I Speak BASIC to My TRS-80™

Aubrey B. Jones, Jr.

A field-tested computer literacy course that introduces students to BASIC language programming.

HAYDEN

I Speak BASIC to My TRS-80™

Aubrey B. Jones, Jr.

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

To Alyce, Aubrey III, and Adrienne

Library of Congress Cataloging in Publication Data

Jones, Aubrey B.
I speak BASIC to my TRS-80.

Summary: A textbook explaining how the computer is used to solve problems, print words, draw pictures, store and retrieve information, and play games.
1. TRS-80 (Computer)--Programming—Juvenile literature. 2. Basic (Computer program language)—Juvenile literature. [1. TRS-80 (Computer)—Programming. 2. Basic (Computer program language) 3. Computers. 4. Programming (Electronic computers)] I. Title.
QA76.8.T18J66 001.64'2 81-13334
ISBN 0-8104-6174-9 AACR2

7 8 9 PRINTING

83 84 85 86 87 88 89 90 YEAR

Contents

PART 1

The Hardware (Or The "Boxes")

What You Will Learn

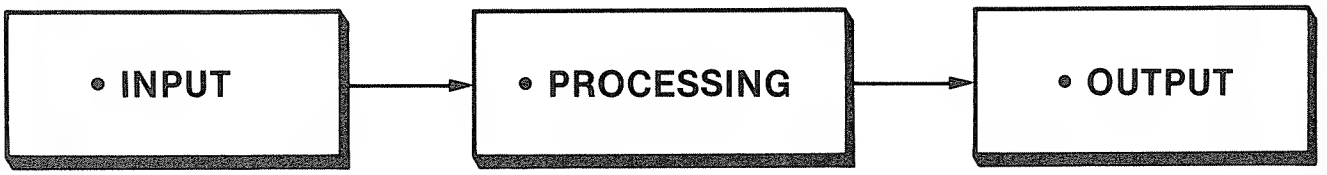
1. That the computer is a valuable tool that can solve problems, print words, draw pictures, store information, retrieve information, compare information, play games, and do many other things to help you in everyday life.
2. That people control computers and that computers cannot think (despite what you might have heard).
3. To identify and explain the basic parts of a computer and relate them to a "box diagram" of a general purpose computer.
4. To identify and explain the function of the basic parts of a TRS-80 microcomputer.
5. To define and explain the terms hardware, software, microcomputer, microprocessor, RAM, ROM, processor, input unit, output unit, memory, and binary.
6. That computers are simple and easy to use; and above all that computers are fun!

Welcome to the World of Computers!

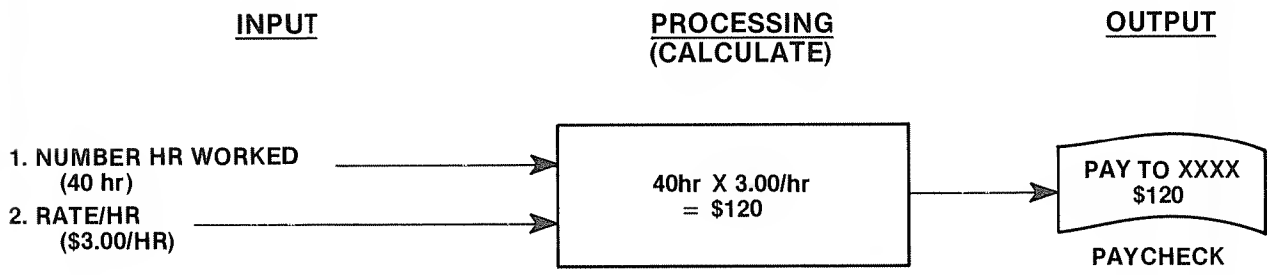
People Control Computers!

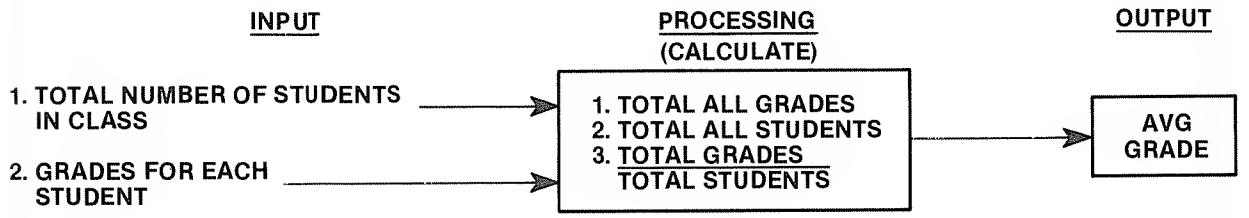
Computers Can't Think!

Typical Data Processing Operation
"Box" Diagram

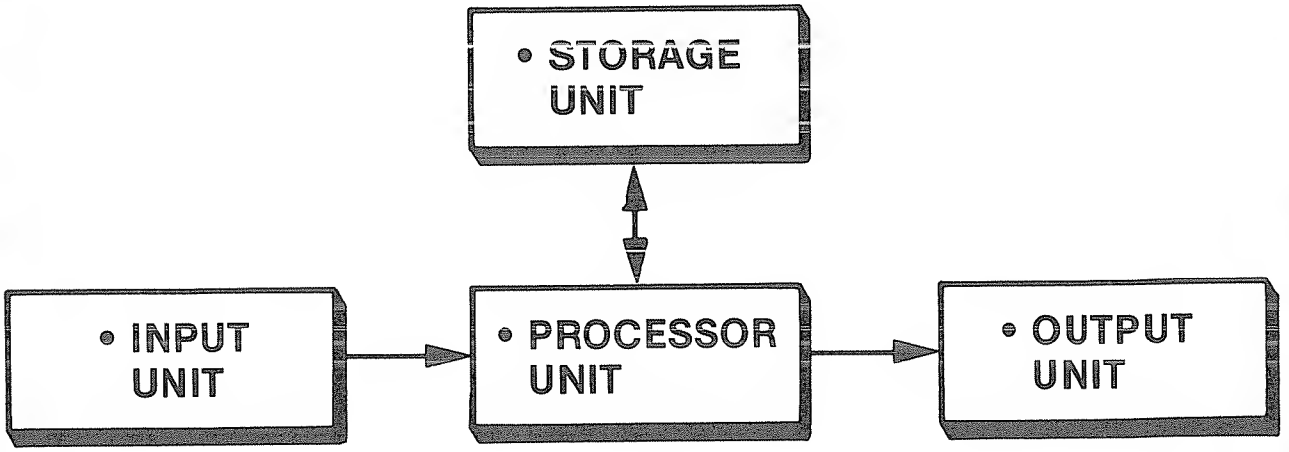


Examples of Data Processing Operation





BOX Diagram
Showing
Basic Parts of a Computer



Stores or Remembers

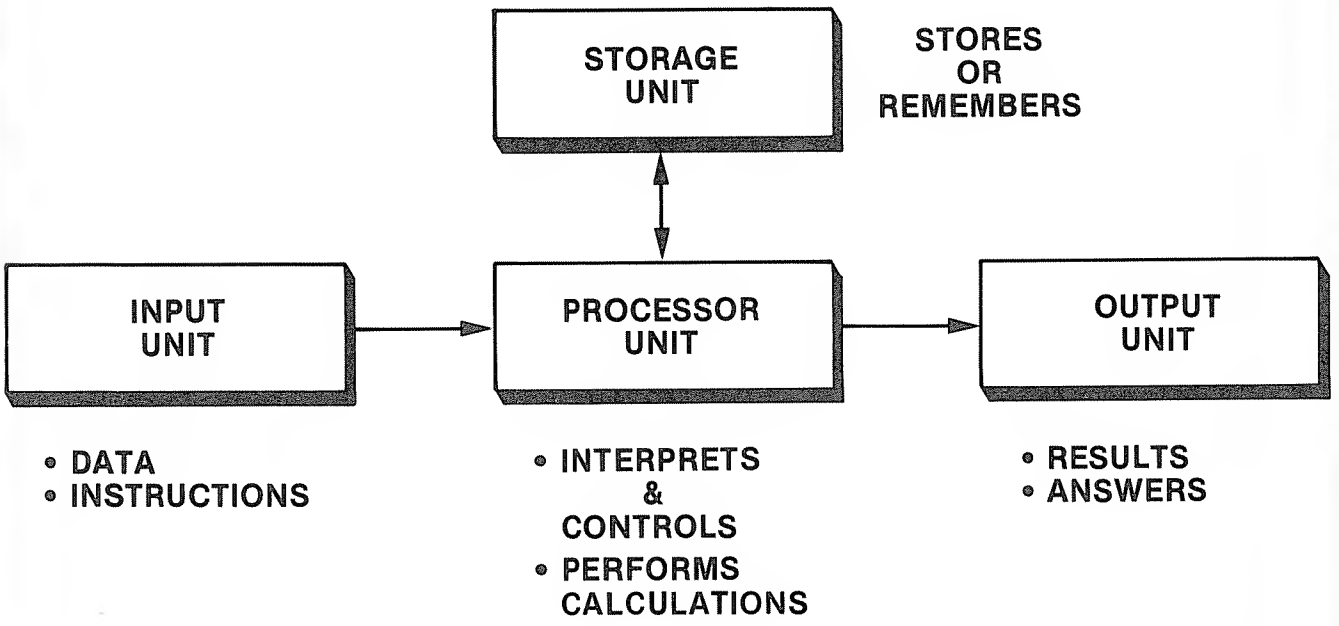
- Storage unit (memory)
  - — Stores both information and instructions until needed (requested)

Interprets, Controls, & Calculates

- PROCESSOR UNIT
  - INTERPRETS (DECODES) INSTRUCTIONS AND REGULATES (CONTROLS) THEIR EXECUTION
  - PERFORMS ALL OF THE CALCULATIONS

Box Diagram of

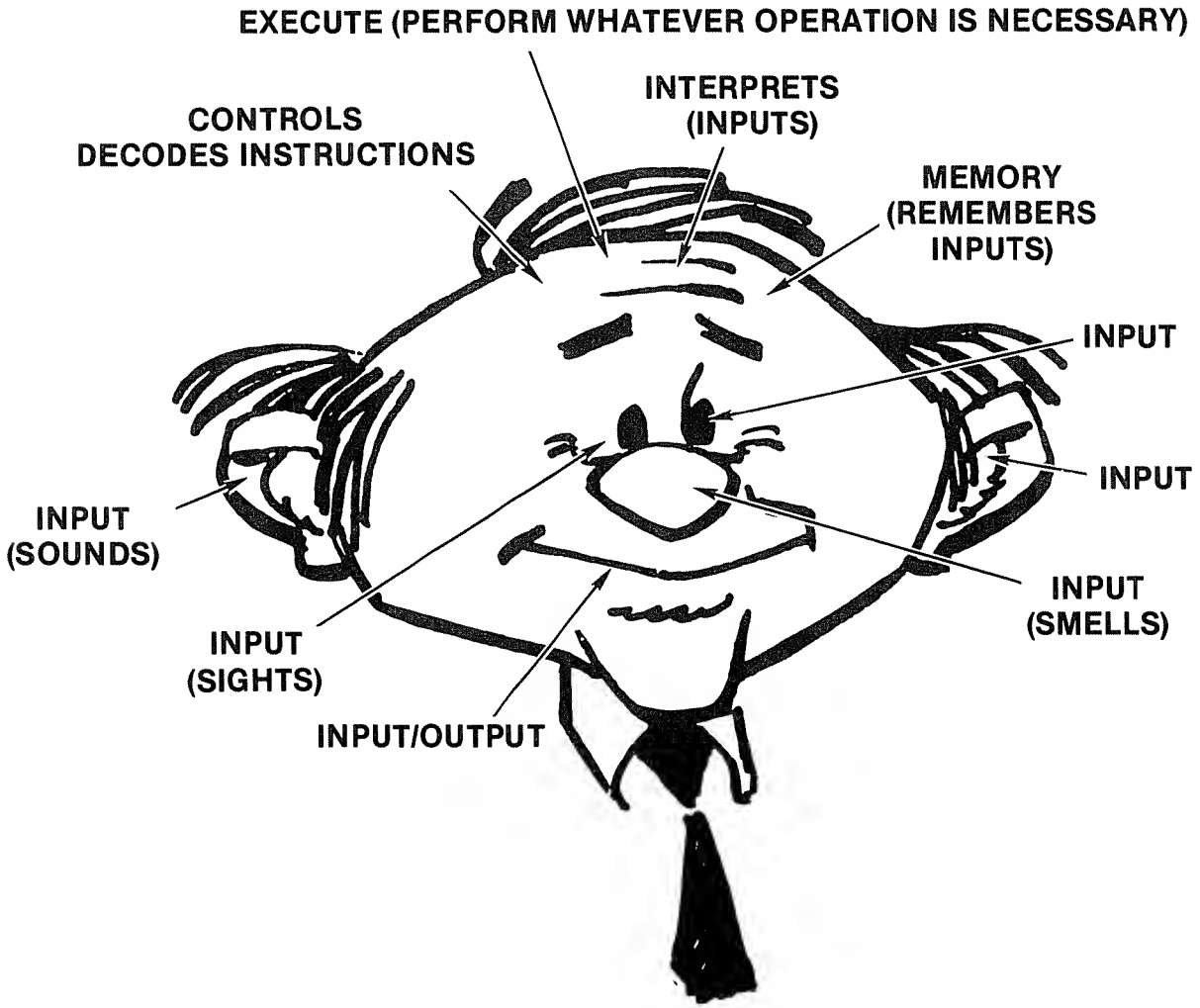
a Basic Computer System



What We Have Learned

- INPUT ⟶ PROVIDES INSTRUCTIONS AND DATA
- STORAGE ⟶ STORES OR REMEMBERS (MEMORY)
- PROCESSOR ⟶ INTERPRETS, CONTROLS, & CALCULATES
- OUTPUT ⟶ PROVIDES ANSWERS AND RESULTS

"Human Computer"
Man Can Think But Computer Can't!

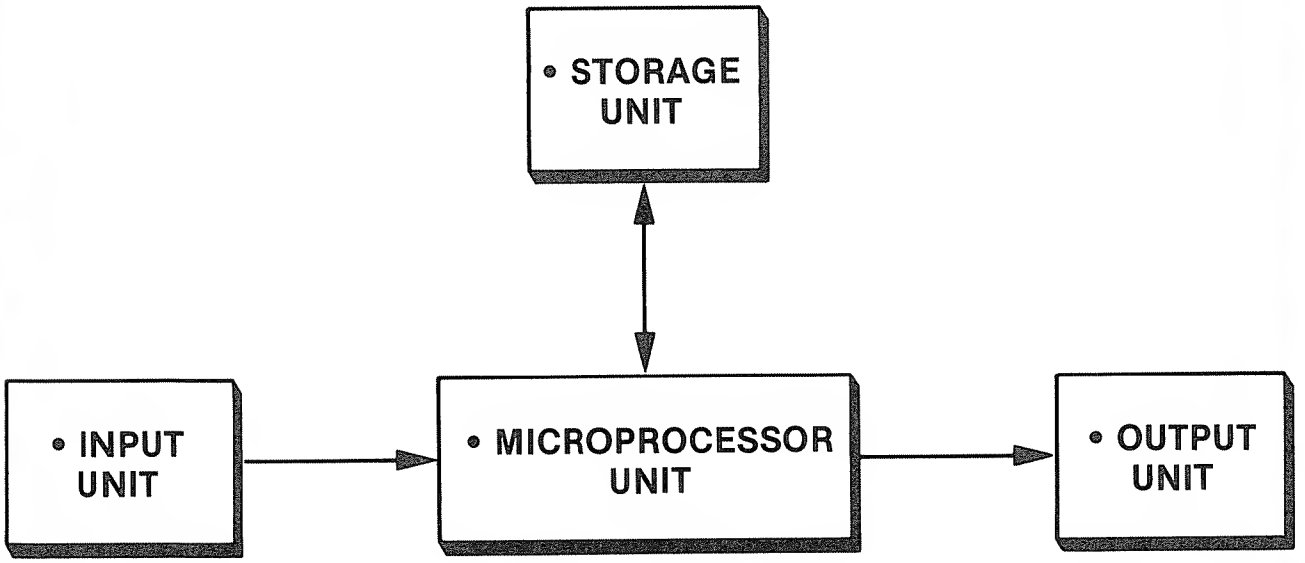


Some Terms You Should Know

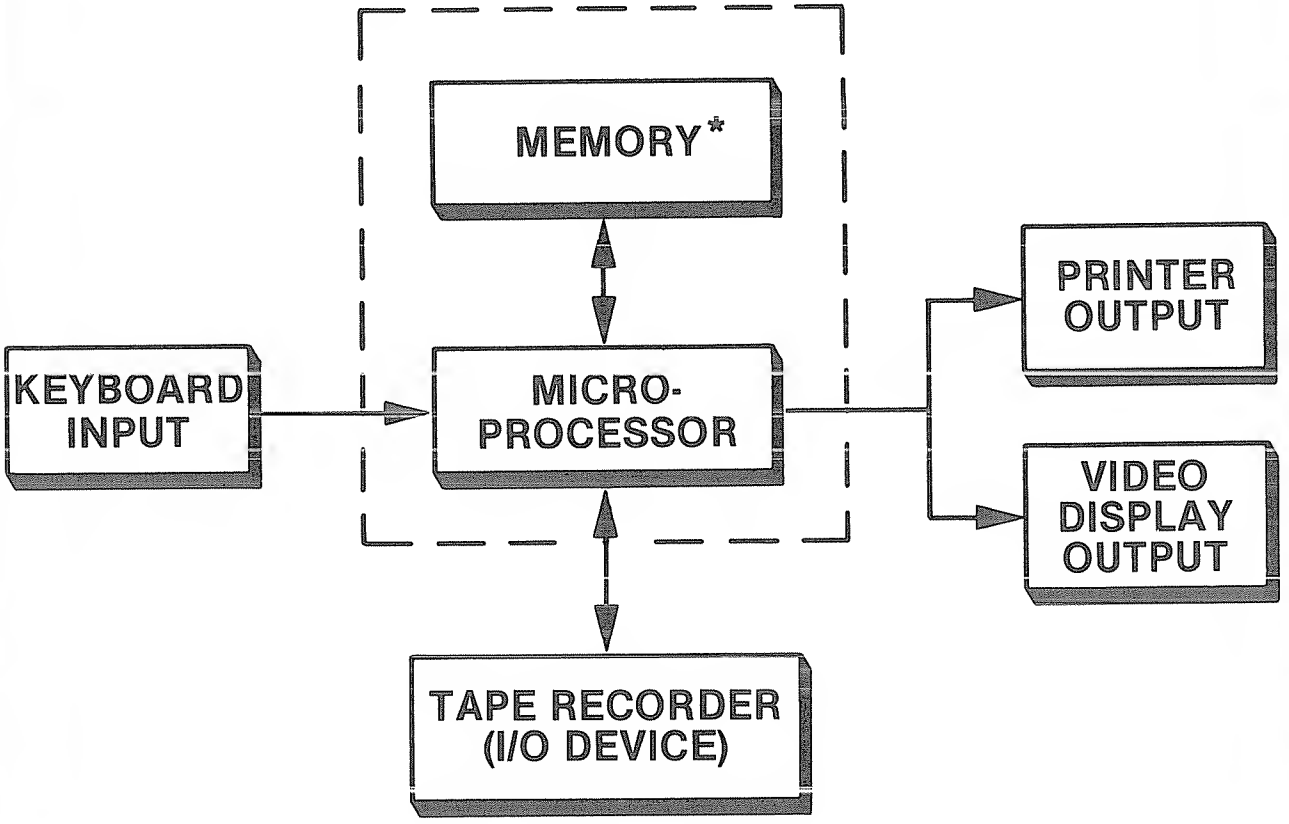
- MICROPROCESSOR
- MICROCOMPUTER
- RAM
- ROM

- MICRO = Very small
- MICROPROCESSOR = Very small processor
- RAM = Random access memory
  - CAN BE changed by the user
  - Information stored in RAM will be destroyed if power fails or is turned off (volatile)
- ROM = Read only memory
  - CANNOT be changed by the user
  - Information stored in ROM is not destroyed if power fails or is turned off (non-volatile)
  - Control program (BASIC interpreter) stored here

Box Diagram of a Microcomputer

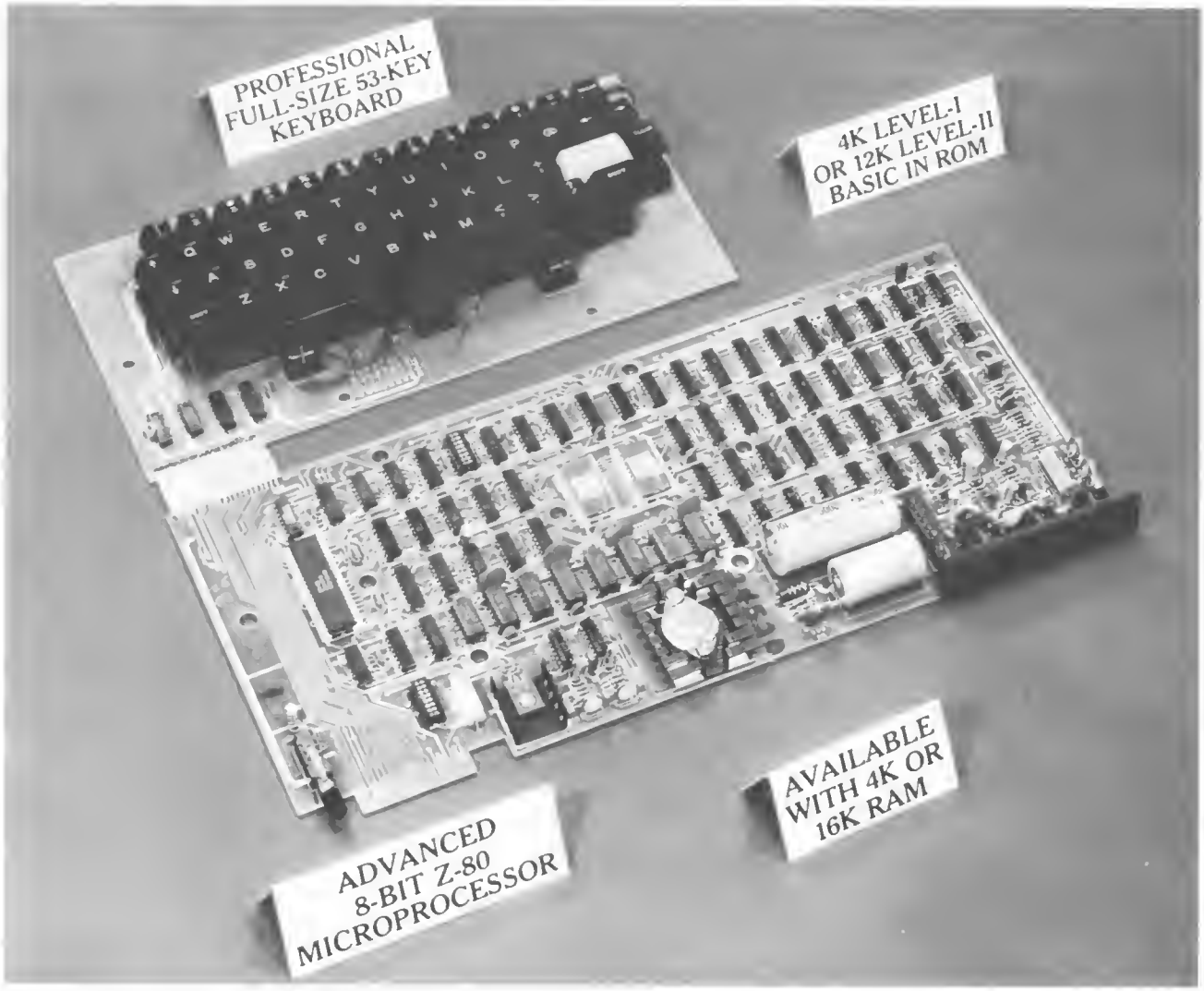


Basic Components of the TRS-80 Computer



*TRS-80 LEVEL II
HAS 12K ROM AND 16K RAM
EXPANDABLE TO 48K

Courtesy of Radio Shack, a Division of Tandy Corporation.



Courtesy of Radio Shack, a Division of Tandy Corporation.

What We Have Learned

DATA PROCESSING OPERATION STEPS:	BASIC COMPUTER PARTS:	MICROCOMPUTER PARTS:
• INPUT →	• INPUT UNIT →	• INPUT UNIT
• PROCESSING →	• PROCESSOR UNIT + MEMORY UNIT →	• MICROPROCESSOR + MEMORY
• OUTPUT →	• OUTPUT UNIT →	• OUTPUT UNIT

PRACTICE 1

Box Diagram of a Computer

1. Draw the BOX DIAGRAM of a BASIC computer.
   a. Label each box with the correct name.
   b. List the functions of each box.

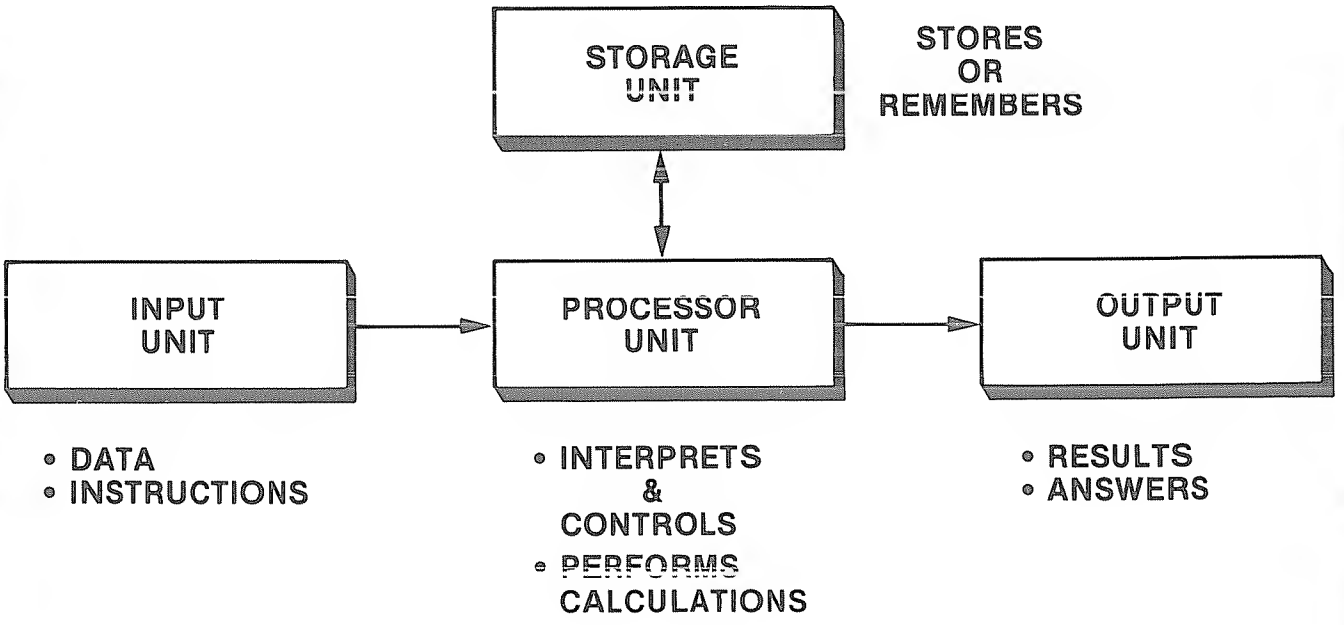
PART 2

The Software (The "Program")

What You Will Learn

1. To define the terms hardware, software, BASIC, binary, and interpreter, and to relate them to computers.
2. That computers speak a foreign language: machine language.
3. How humans talk to computers via a programming language called BASIC.
4. To identify the principal parts of a BASIC program.
5. To identify and explain the purpose of all the keys on the TRS-80 keyboard.
6. How to connect and power up a TRS-80 microcomputer.

Box Diagram of a Basic Computer System



More Terms You Should Know

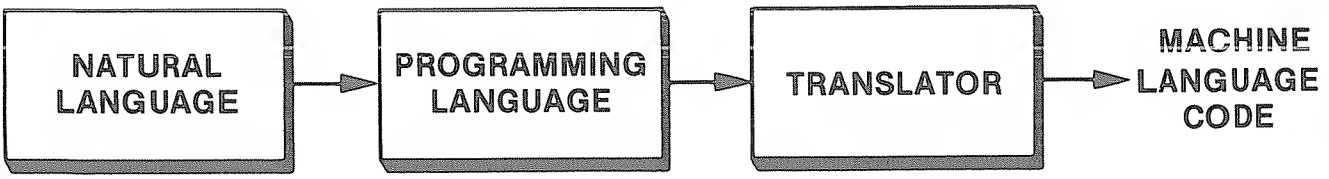
- HARDWARE
  - — THE COMPUTER AND COMPUTER RELATED EQUIPMENT (THE BOXES)

- SOFTWARE
  - — THE INSTRUCTIONS FOR THE COMPUTER (THE PROGRAM)

Computers Speak a Foreign Language! (No Speak English, French, German Spanish, or Any Other Natural Language)

- COMPUTERS SPEAK IN *MACHINE* LANGUAGE
  - MACHINE LANGUAGE IS A FORM OF *BINARY* CODING
  - BINARY IS A WORD DENOTING "TWO"
  - MACHINE LANGUAGE USES TWO BASIC SYMBOLS: "Ø" AND "1"

How Humans Talk to Computers



- ENGLISH — THINK
- BASIC — WRITE
- INTERPRETER — INTERPRET
- MACHINE — EXECUTE

- **BASIC**

  (Beginner's all-purpose symbolic instruction code)

  — **Popular programming language for writing instructions to the computer**

- **INTERPRETER**

  — **Translates BASIC into machine code**
  — **(You really don't have to know anything about an interpreter since it is used automatically when you run a BASIC program)**
  — **Located in the ROM in TRS-80**

To Program You Must Learn the Language First!

A Comparison between English and BASIC

ENGLISH LANGUAGE

- Words
  - — Used to make sentences
- Sentences
  - — Used to make paragraphs
- Paragraphs
  - — Lengths vary
- Commands
  - — Can be one word
  - — e.g., STOP! HALT!
- Sentence Numbers
  - — Optional (seldom used)

BASIC PROGRAMMING LANGUAGE

- Key Words
  - — Used to make statements
- Statements
  - — Used to make programs
- Programs
  - — Lengths vary
- Commands
  - — Executed immediately
  - — e.g., NEW, LIST, RUN
- Line Numbers
  - — Must be used for each statement

Learning a New Vocabulary

Here Are the Key Words and Commands You'll Learn:

KEY WORDS

- PRINT
- END
- LET
- INPUT
- GO TO
- IF --- THEN
- REM
- STOP
- FOR --- NEXT
- READ-DATA

COMMANDS

- NEW
- LIST
- RUN
- CONT

Commands vs. Statements

COMMANDS

—Executed as soon as you type them and press ENTER

STATEMENTS

—Put into programs and are only executed after you type the command RUN and press ENTER

A BASIC Program

	LINE NUMBER	KEY WORD	OTHER PART OF THE STATEMENT	"LOOK AT" REQUEST*
1st STATEMENT	1Ø	PRINT	"HELLO THERE"	ENTER
2nd STATEMENT	2Ø	PRINT	"YOUR NAME"	ENTER
3rd STATEMENT	3Ø	END		ENTER
COMMAND	RUN			ENTER

*Pressing the ENTER key tells the computer to "LOOK AT" (and store) what you have just typed. You must press this key after each statement or command.

Line Numbers

— Serve as a guide to the computer in running the program.
— Tell the computer in what order it should carry out your instructions.
— Computer will start executing at lowest numbered line unless told to start elsewhere.
— Normally are multiples of 5's, 10's, or some other multiples to leave space for inserting new program lines between old one.
— Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4, etc.), don't do it!

Key Words

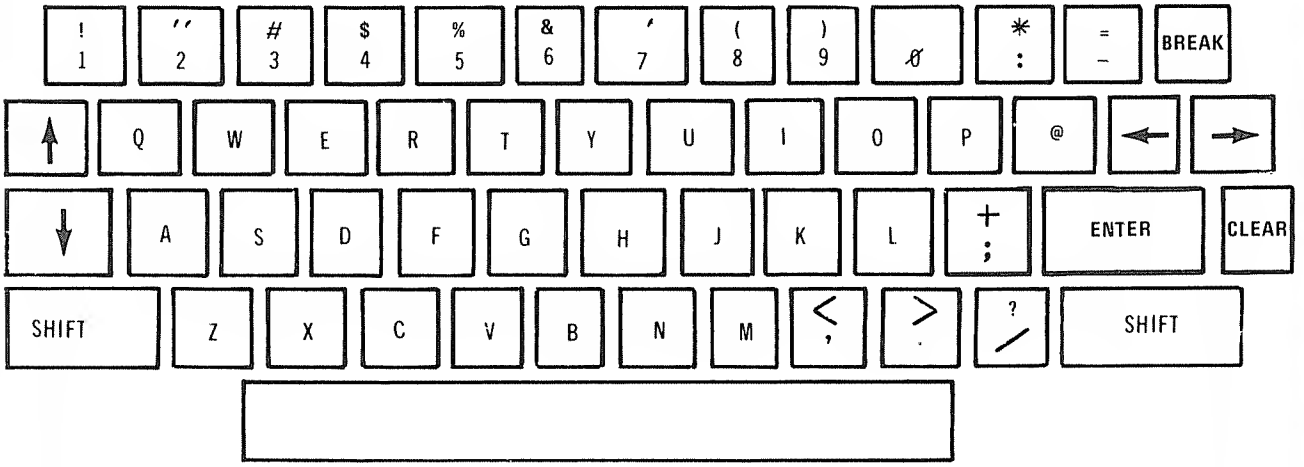
- Never used alone
- Need line number
- Always part of a BASIC statement that has some other part to it*
- Executed only after command RUN is typed and [ENTER] key is pressed

*To the purist, we know that key words like END and STOP can be used alone; but you still need line numbers, and you must type RUN and press [ENTER] to execute.

What We Have Learned

- Key words
  - — Used to make statements
- Statements
  - — Must have line numbers and key words
  - — Used to make programs
- Programs
  - — May vary in length
- Commands
  - — Executed as soon as you type them and press ENTER

TRS—80 KEYBOARD



Special Function Keys on the Keyboard

KEY	FUNCTION
BREAK	• Stops execution of a program. To continue execution, type CONT and press ENTER.
CLEAR	• Clears display. Also, if 32 character per line format (large characters) is being used, pressing this key returns display to 64 character per line format. (To get prompt > back on display press ENTER.)
ENTER	• Causes computer to "look at" line you just typed in and to act accordingly. This key must be pressed each time you want to enter a line from the keyboard.
SHIFT	• Some keys have two characters printed on them. Use this key to type upper characters like quotes (").
←	• Back spaces the cursor and deletes last character typed in.
→	• Moves cursor to the next tab stop. Tab stops are at position 0, 8, 16, 24, 32, 48, and 56.
↓	• Moves cursor down to next physical line on display.
↑	• An arithmetic operator for exponentiation. For example, if you wanted to type 2^3, you would type 2↑3 instead.

TRS-80 Power-Up Rules

1. Make certain system is connected properly.
2. If tape recorder is connected, it should be in STOP mode.
3. Press power buttons on video display and at the back of the keyboard (give video tube a few seconds to warm up).
4. Memory size? (should appear on screen)
5. Press [ENTER] key on the keyboard.
6. Radio Shack Level II BASIC
   Ready
   } (Will appear on screen)
7. You are now ready to use Level II BASIC!
8. Type in the word NEW and press [ENTER].
9. NOTE!
   If your TRS-80 "freezes up" (you can't type anything), you can press [RESET] button on the back of the keyboard on the left side to return to the memory size? question.

Getting It Together

- STEP 1 — WRITE YOUR PROGRAM
- STEP 2 — GET THE COMPUTER READY
- STEP 3 — ENTER YOUR BASIC PROGRAM
- STEP 4 — RUN YOUR PROGRAM
- STEP 5 — SIGN OFF

PRACTICE 2

Becoming Familiar with the TRS-80

Become familiar with the TRS-80 by doing the following (you should actually go through every step.):

1. Power up (turn on) TRS-80 using TRS-80 power up rules (see page 33).
2. How many power buttons did you have to press? _____
3. Where were the power buttons located? _____
4. Where is the reset button located? _____
5. Locate the cursor.
   a. Move it to the right using [→]
   b. Move it to the left using [←]
6. Locate the [SHIFT] key.
   a. Type some letters using the [SHIFT] key.
   b. Press same keys without using [SHIFT] key.
7. Locate the [ENTER], [BREAK], and [CLEAR] keys,
   You will learn to use these keys later.

PART 3

Your First Computer Program

What You Will Learn

1. To enter and run your first BASIC program.
2. To explain the purpose and use of the following BASIC commands: LIST, NEW, RUN.
3. To explain the purpose and use of the following key words: PRINT, PRINT (for spacing), REM, END.
4. To explain the purpose and use of the following special function keys: [BREAK], [CLEAR], [ENTER], [SHIFT], [←] (back space).
5. To explain the purpose and use of the following miscellaneous points: > prompt, _ cursor, " " (quotes), line numbers, reset button, power-up rules.

Special Function Keys on the Keyboard

KEY	FUNCTION
BREAK	• Stops execution of a program. To continue execution, type CONT and press ENTER.
CLEAR	• Clears display. Also, if 32 character per line format (large characters) is being used, pressing this key returns display to 64 character per line format. (To get prompt > back on display, press ENTER.)
ENTER	• Causes computer to "look at" line you just typed in and to act accordingly. This key must be pressed each time you want to enter a line from the keyboard.
SHIFT	• Some keys have two characters printed on them. Use this key to type upper characters like quotes (").
←	• Back spaces the cursor and deletes last character typed in.
→	• Moves cursor to the next tab stop. Tab stops are at position 0, 8, 16, 24, 32, 48, and 56.
↓	• Moves cursor down to next physical line on display.
↑	• An arithmetic operator for exponentiation. For example, if you wanted to type 2^3, you would type 2↑3 instead.

TRS-80 Power-Up Rules

1. Make certain system is connected properly.
2. If tape recorder is connected, it should be in STOP mode.
3. Press power buttons on video display and at the back of the keyboard (give video tube a few seconds to warm up).
4. Memory size? (should appear on screen).
5. Press [ENTER] key on the keyboard.
6. Radio Shack Level II BASIC
   Ready
   } (Will appear on screen)
7. You are now ready to use Level II BASIC!
8. Type in the word NEW and press [ENTER].
9. NOTE!
   If your TRS-80 "freezes up" (you can't type anything), you can press [RESET] button on the back of the keyboard on the left side to return to the memory size? question.

Typical Display Readout

```
1Ø          PRINT          "HELLO THERE"

2Ø          PRINT          "YOUR NAME"

3Ø          END

RUN
```

Writing Your First Computer Program

YOUR ACTION

1. Before you start typing your program, always type NEW and press the ENTER key.
2. Type the line exactly as shown: ⟶
3. Use SHIFT key to type the upper characters like the quotation marks (") and the exclamation point (!).
4. Do *not* press ENTER key yet!
5. Go back and examine your typed line *very carefully*. Did you make a mistake? If you did, just use the backspace key ← to erase a character.
   (Note: If you made a mistake at the beginning of the line, you will have to erase your way back to that point and then retype the rest of the line.)
6. Is everything OK? If it is, you can press ENTER. (This tells the computer to "look at" what you just typed in).
7. The prompt > should appear. The computer is saying, "Its your turn... I'm waiting for you."

DISPLAY

```
1Ø PRINT "HELLO THERE NAME!" _
```

Ⓐ

```
1Ø PRINT "HELLO THERE NAME!"
>_
```

NOTE

Ⓐ Insert student's name

Go to next page

Common Errors

- Missing quotes (”)
- Too many quotes
- Forgot the key word PRINT
- Forgot the line number
- Forgot to press ENTER
- Used the character “O” for the number “ZERO” (Ø).

(Note: A slash is used to help you to recognize a zero.)

Writing Your First Computer Program — Almost? (Errors)

PROBLEM (You Forgot to Follow Instructions)	SOLUTION
1. MISSING QUOTES (") — You forgot to enclose everything after the word PRINT in quotation marks. (If you want something printed, don't forget the quotation marks!)	• If you have already pressed ENTER, you must retype the entire line to correct your error. Here's how you do it:
2. TOO MANY QUOTATION MARKS — You typed too many. (That won't work either!)	• Type in the same line number you wish to change (1Ø in this case). That is, if you want the computer to replace that line with the corrected line.
3. FORGOT THE KEY WORD PRINT — You forgot to type PRINT. (How will the computer know you want to print if you don't tell it to print?)	• Next, retype the line exactly as shown on previous page. (But follow directions this time, Dummy!) • Then, check line over for errors.
4. FORGOT TO TYPE THE LINE NUMBER (1Ø) — Line numbers tell the computer where to start. The computer always starts executing from the lowest numbered line unless you tell it to start elsewhere. (We will show you how to tell the computer to start at another line later. Keep the faith!)	• If everything is OK, don't forget to press ENTER ! When you press ENTER it tells the computer to "look at" what you just typed and to act accordingly.

Read this page if you had any errors! Then correct your errors before going to the next page!

Executing Your Program

YOUR ACTION	DISPLAY
1. Tell the computer to execute or run your program. The command for this is simple: RUN.	
2. So type *RUN* and press ENTER.	
3. If you made no mistakes, the display will read: →	HELLO THERE NAME!
4. If it did not work, try again (i.e., check your program for errors).	
5. If it did work, let out a yell, "HEY, I CAN DO IT TOO!"	

Go to next page (if you completed this one OK)

Expanding Your Program

YOUR ACTION	DISPLAY
1. You now have a program in the computer. (Unless you turned it off. If you did, retype line as shown): →	1Ø PRINT "HELLO THERE NAME"
2. Type in line 2Ø *exactly* as shown: →	2Ø PRINT "I'M GOING TO MAKE YOU A SUPERSTAR!"
3. Check your new line (2Ø) *very carefully*, especially the quotation marks.	
4. Everything OK? Press ENTER. (Remember, always press ENTER if you want the computer to look at what you typed.)	
5. Let's run your program. Type RUN and press ENTER.	
6. If you did it right, the screen will read:	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR!
7. If it did not work, check your program for errors.	

Go to next page

Using the PRINT Statement for Spacing

YOUR ACTION	DISPLAY
1. Look at your video display. Would you like more space between the two lines? OK, this is how you do it.	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR!
2. Type in a new line as shown ⟶ and then press ENTER.	15 PRINT
3. Now type *RUN* and ENTER ⟶	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR!
4. WOW! A *PRINT* "nothing" puts a space between what you told the computer to print in Lines 1Ø and 2Ø.	
5. Observe that the PRINT statement (Line 15) was placed between Lines 1Ø and 2Ø. Since you were smart enough to number your lines by 1Ø's, it was much easier to modify your program. (That's because you left room to insert new lines between the old ones.) Although it is perfectly legal to number program lines more closely (like 1, 2, 3, 4), don't do it.	

Go to next page

Inserting Remarks into a Program (But Not Printing Them Out)

YOUR ACTION

1. Another important statement is REM, which stands for remark. It is often convenient to insert remarks into a program. The main reason for inserting remarks is so you or someone else can refer to them later and know what the program is for and how it is used.
2. When you tell the computer to execute the program by typing RUN and [ENTER], it will skip right over any number line that begins with the statement REM. The REM statement will have no effect on the program. (Let's see about that!)
3. Type Line 5 exactly as shown and then [ENTER] (*'s are just for decoration).
4. Type RUN and press [ENTER].
5. It is the same as before (REM statement was not printed).

Go to next page

DISPLAY

```
5 REM *THIS IS MY FIRST COMPUTER PROGRAM*
```

```
HELLO THERE NAME!
I'M GOING TO MAKE YOU A SUPERSTAR!
```

Listing Your Program
(Looking At Your Program to See What It Contains)

YOUR ACTION*

DISPLAY

1. **To list your program is easy. The command is LIST.**
2. **Now you type LIST and [ENTER]:** →

```
 5 REM * THIS IS MY FIRST COMPUTER PROGRAM*
1Ø PRINT "HELLO THERE NAME!"
15 PRINT
2Ø PRINT "I'M GOING TO MAKE YOU A SUPERSTAR"
```

3. **You can call for a listing of your program any time the prompt > appears on the screen.**
4. **Also, you might only want to list one line. Type LIST 2Ø and press [ENTER] and the screen will display:**

```
2Ø PRINT "I'M GOING TO MAKE YOU A SUPERSTAR"
```

5. **You might also want to list several program lines, starting at one line and ending at another. For example, type List 1Ø - 2Ø and [ENTER].**

```
1Ø PRINT "HELLO THERE NAME!"
15 PRINT
2Ø PRINT "I'M GOING TO MAKE YOU A SUPERSTAR!"
```

***Press [CLEAR] key so you can start with a clean display. Then press [ENTER], so prompt > will return to screen.**

Go to next page

Ending Your Program

YOUR ACTION	DISPLAY
1. The end of a program is the last statement you want the computer to execute. Most computers require you to place an END statement after this point, so the computer will know it is finished. However, the TRS-80 does *not* require an END statement. (Other computers might require it though.)	
2. Let's add an END statement to your program. Type and ENTER : ⟶	99 END
3. Now type RUN and ENTER .	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR!
4. No change from before! The program ended, but it did not print "END."	
5. Let's make it print THE END. (How do we do that?)	
6. Oh, I remember! We need a PRINT statement. So let's try it. Type and ENTER : ⟶	98 PRINT "THE END"
7. Now RUN your program.	
8. IT WORKED AGAIN! (If not, check the program.)	HELLO THERE NAME! I'M GOING TO MAKE YOU A SUPERSTAR! THE END
9. Note that there is no space between THE END and the line above it. Why? (Because you did not tell the computer to put a space between them!)	

Learned in This Session

COMMANDS*	KEY WORDS**	MISCELLANEOUS	SPECIAL FUNCTION KEYS
• LIST — LIST MM • NEW • RUN — RUN MM	PRINT "MESSAGE" PRINT (SPACE) REM END	> PROMPT – CURSOR " " QUOTATION MARKS LINE NUMBERING RESET • KEYBOARD LAYOUT • TRS-80 POWER-UP RULES • READY	BREAK CLEAR ENTER SHIFT ← BACKSPACE → MOVES CURSOR TO NEXT STOP ↓ MOVES CURSOR DOWN

* Executed as soon as you type them and press ENTER

** Used to make statements. Statements are executed after you type RUN and press ENTER

NOTE: If you don't understand everything on this page, stop!
Go back over this session until you understand it thoroughly!
MM = Any line number (e. g., 1Ø, 2Ø, 3Ø, etc.)

Assignment* 3-1

I. WRITE* A PROGRAM TO PRINT ON SEPARATE LINES
   - A. Your Name
   - B. Your Entire Address
   - C. Your Telephone Number

II. EXPAND* YOUR PROGRAM TO INCLUDE THE FOLLOWING:
   - A. Remark Statement to Describe Your Program
   - B. Spacing between Each of the Lines Displayed (Printed)
   - C. Include an End Statement

III. TYPE YOUR PROGRAM AND ENTER

IV. RUN YOUR PROGRAM

V. LIST YOUR PROGRAM

* WRITE YOUR PROGRAM ON PAPER AND GET IT CHECKED BY YOUR TEACHER FIRST.

PRACTICE 3

Writing and Running Your First Program

1. Write a program to PRINT the following:
   a. Your name (first and last)
   b. Your school's name
   c. Your teacher's name
2. ENTER and RUN it.

PRACTICE 4

Inserting Remarks and Spacing into Your Program

1. If you have erased the program from Practice 3, rewrite the program and do the following: (If you still have the program from Practice 3 in the computer, you do not have to rewrite the program.)
   a. Add a new program line with a remark statement to your program (any remarks you want to make).
   b. Have the computer insert one space between your name and your school's name in the output on the display (that is, you add the necessary program line).
   c. Have the computer insert two spaces between your school's name and your teacher's name in the output on the display.

PRACTICE 5

Listing and Ending Your Program

1. Rewrite the program from Practice 4 and do the following (Again, if you have the program in the computer, you don't have to rewrite it. But in case you don't know what is in the computer, just type NEW and rewrite the program.):
   a. Add an END statement to tell the computer it is the end of your program.
   b. Add a statement to have your computer PRINT "THE END."
   c. RUN your program.
2. List your program.
   a. How large is your program now? (How many lines?)
   b. Copy the program in your notebook.

PART 4

More Programming Tools

What You Will Learn

1. To enter and run more BASIC programs: mathematical programs, area of rectangle program.
2. To explain the order of mathematical operations using the M.D.A.S. rule.
3. To explain the purpose and use of the keyword: LET.
4. To explain the purpose and use of the BASIC mathematic operators: multiply (*), divide (/), add (+), subtract (—), exponentiate or raising a number to a power (↑).
5. To explain the function and use of commas, semicolons, and print zones.
6. To list and identify variables that can be used with TRS-80 Level II BASIC.

Review of Part 3

COMMANDS*	KEY WORDS**	MISCELLANEOUS	SPECIAL FUNCTION KEYS
• LIST — LIST MM • NEW • RUN — RUN MM	PRINT "MESSAGE" PRINT (SPACE) REM END	> PROMPT – CURSOR " " QUOTATION MARKS LINE NUMBERING RESET • KEYBOARD LAYOUT • TRS-80 POWER-UP RULES • READY	BREAK CLEAR ENTER SHIFT ← BACKSPACE → MOVES CURSOR TO NEXT STOP ↓ MOVES CURSOR DOWN

* Executed as soon as you type them and press ENTER

** Used to make statements. Statements are executed after you type RUN and press ENTER

NOTE: If you don't understand everything on this page, stop!
Go back over this session until you understand it thoroughly!
MM = Any line number (e. g., 1Ø, 2Ø, 3Ø, etc.)

Math Operators

= (Equal)

+ (Add)

– (Subtract)

* (Multiply)

/ (Divide)

↑ (Exponentiation)

(↑) means raising a number
to a power like 2^2, 2^3, or 2^4

Order of Arithmetic Operations

- Multiply ⟶ Divide ⟶ Add ⟶ Subtract
  (Left to Right)
  — "My Dear Aunt Sally"

- If Parentheses are used
  — Innermost level operations first
  — Then next level out
  — M.D.A.S. order inside parentheses

Order of Operations Example — (Without Parentheses)

- If there are no parentheses, the computer performs operations by going from left to right doing exponentiation operations (↑) first. Then (*) and (/) are done in order from left to right and finally (+) and (−) are done in order from left to right. (Remember M.D.A.S.!)

Example:

4 + 5 * 4 ↑ 3 − 4/2 =
4 + 5 * [64] − 4/2 =
4 + [320] − 4/2 =
4 + 320 − [2] =
[324] − 2 = [322]

In-Class Exercise 4-1

You Try Some Now (Without Parentheses)

1) 2 ↑ 3 + 4 * 5 − 4/2 * 5 = _____
2) 14 − 2 * 2 + 6 − 2 * 3 * 2 = _____
3) 14/2 * 3 − 2 ↑ 3 + 4 = _____

Now try some with parentheses

1) 6 + (9 * 2) = _____
2) (6 + (9 * 2)) * 5 = _____
3) 3 * ((4 + (6 * 2)) * (9/3 − 1)) = _____

A computer is not required here, but it could be used to check the answers. You don't need a line number for calculator mode. Simply type PRINT and the calculations you want done. Example: If you wish to multiply 2 asterisk 3, simply PRINT 2 * 3 and press RETURN. The answer (6) will be displayed.

Order of Operations Example — (With Parentheses)

- If there are parentheses, the computer starts at the inner pair of parentheses and converts everything to a single number. Then the computer repeats the process with the next pair of parentheses working "inside" out.

Example:

((6 + 4) * 2) /4 =

(1Ø * 2) /4 =

2Ø /4 = 5

Tips on Using Parentheses — Summary

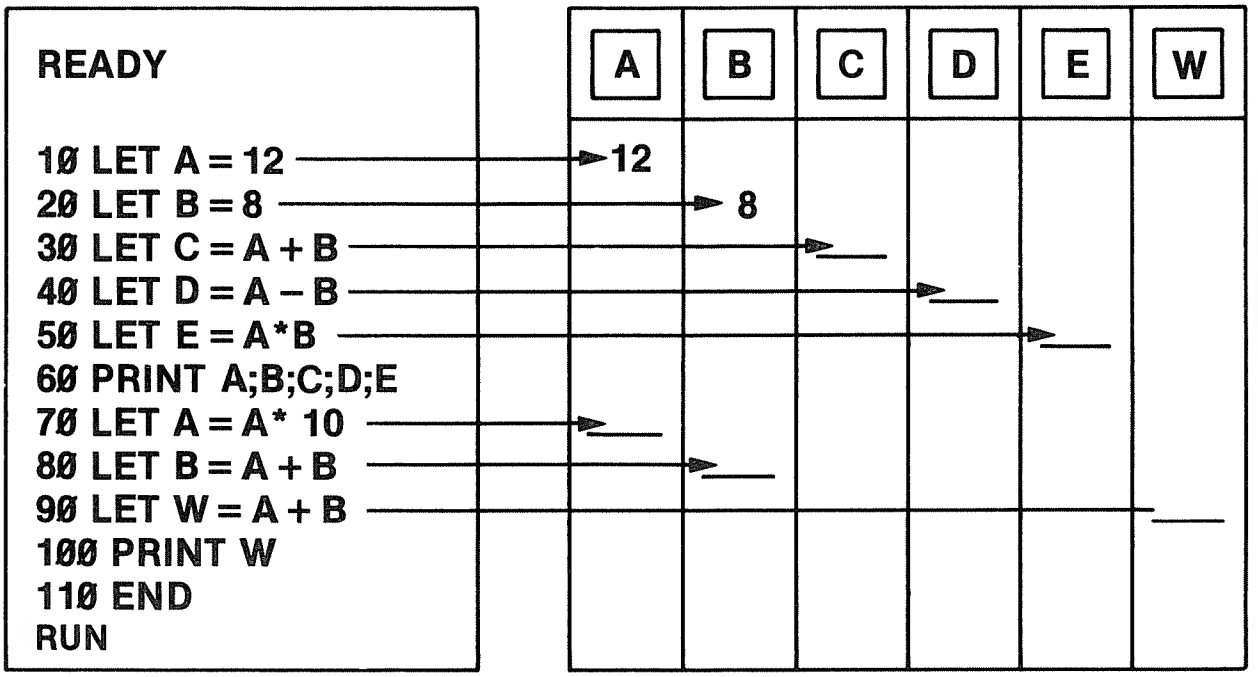
- When in doubt, use parentheses. They can't do any harm!
  - — Use parentheses around operations you want performed first
- Make sure that every left parenthesis has a matching right parenthesis
  - — Count them to be sure!
- Order of Operations
  - — Inner most pair of parentheses first (M.D.A.S. rule inside parentheses)
  - — Then work "inside" out
  - — In case of a "tie," computer starts to the left and works right doing exponentiation (↑) and the M.D.A.S. rule.

Variable Names Used with TRS-80 Level II BASIC

- Must begin with a letter (A–Z)
  - — May be followed by another letter
    or
  - —May be followed by a digit (Ø–9)
- Some examples of variable names include:
  - — A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.
  - — A1, A2, B1, B2, C3, C5, D9, N9, P4, Q1, R6, Y7
  - — AA, AZ, GP, MU, ZZ, BB, XY, LL, FG, LE, RE

    (You get the picture! Using the above combinations, you can use approximately 9ØØ variable names.)
- There are some words with special meaning in the BASIC language and they *cannot* be used as variable names.
  - — The complete list of reserved words, which cannot be used in variable names, appears in Appendix A of the Radio Shack Level II Basic Reference Manual.

In-Class Exercise 4-2
(Assigning Numeric Values to Variables)



Basic Program for a Mathematical Operation

Line No.	Key Word[1]	Other Part of Statement	
1Ø	LET	X = 5	ENTER
2Ø	LET	Y = 12	ENTER
3Ø	LET	Z = X*Y	ENTER
4Ø	PRINT	Z	ENTER
99	END		

RUN

(1) LET is an optional key word for TRS-80 BASIC. Some computers require you to use LET however. Beware of this if you use another computer.

Analysis of the BASIC Program for a Mathematical Operation

Line No.	Statement	Meaning to Computer
1Ø	LET X = 5	Assign a value of 5 to variable X
2Ø	LET Y = 12	Assign a value of 12 to variable Y
3Ø	LET Z = X*Y	Take the values of X and Y, multiply them together, and assign the resulting value to the variable Z
4Ø	PRINT Z	Print the value of Z (which is 6Ø in the example)
99	END	END PROGRAM
RUN		EXECUTE PROGRAM

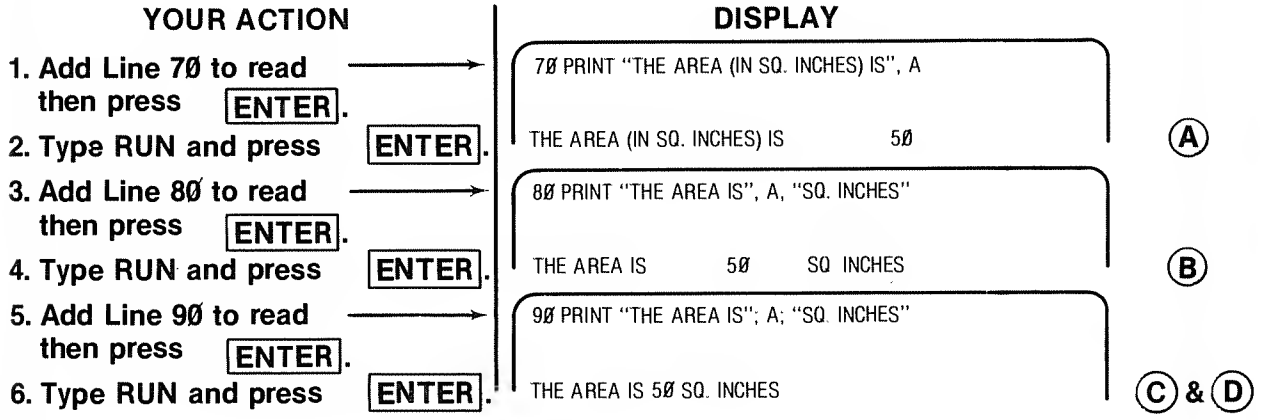
A BASIC Mathematical Program —

Area of Rectangle

YOUR ACTION	DISPLAY
1. Type NEW	
2. Type and ENTER →	1Ø REM * AREA OF A RECTANGLE PROBLEM * 2Ø REM * AREA (A) = LENGTH (L) X WIDTH (W) * 3Ø LET L = 1Ø 4Ø LET W = 5 5Ø LET A = L*W 6Ø PRINT A
3. Type RUN and press ENTER	RUN 5Ø

NOTE THAT WE SAID IN LINE 6Ø PRINT A. There were no quotes around the letter A because we wanted the computer to PRINT the *value* of A. If we wanted the computer to PRINT the exact word or letter, we would put quotes around the word or variable.

Area of Rectangle Program Modified



Notes:

Ⓐ **Comma in Line 7Ø told the computer to print two separate items on the same line.**

Ⓑ **Commas in Line 8Ø told the computer to print three separate items on the same line.**

Ⓒ **A semicolon automatically inserts one space between two items it is separating if the two items include a variable and a "message." (This is unique to TRS-80. In standard BASIC programming, a semicolon tells the computer to PRINT output close together with no spacing. So beware!)**

Ⓓ **LIST your program when you finish. RUN your program several times and note that you have printed your answer four different ways.**

Assignment 4-1

1. Write a Program to Find Area of a Triangle
   A. GIVEN: A = ½ BH WHERE B = 5, H = 1Ø
   B. Include Remarks Statement
   C. Have Program PRINT "THE AREA = " (Your Answer) "SQ. FT."

2. Write a Program to Find the Volume of a Rectangular Solid
   A. GIVEN V = L*W*H, L = 5, W = 1Ø, H = 2
   B. Include Remarks Statement
   C. Have Program PRINT "THE VOLUME = " (Your Answer) "CUBIC IN."

3. Given the formula for converting Fahrenheit to Celsius as follows:

$$C° = (F° - 32) \times (5/9)$$

   A. Write and RUN a program that converts 75° Fahrenheit to Celsius.
   B. Change the value of F from 75° to 45° and RUN the program again.

4. Given the formula for converting Celsius to Fahrenheit as follows:

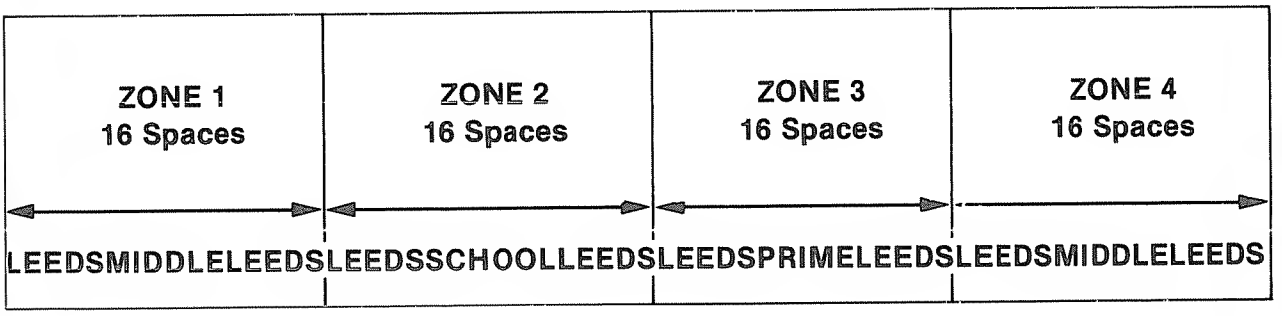
$$F° = 9/5 \times C° + 32$$

   A. Write and RUN a program to find F if C is 20°.
   B. Change the value of C from 20° to 35° and RUN the program to find F.

Summary — Mathematical Operations

- LET is an optional key word when using TRS-80 BASIC.
  — Other computers using BASIC might require use of LET, so beware!
- 1Ø PRINT A: Tells computer to print the value of A
  — Whereas 1Ø PRINT "A": Tells computer to print letter A (because the computer will print anything within quotes).
- A comma in a PRINT statement tells the computer to leave several spaces between items separated by the commas.
- A semicolon inserts one space between two items it is separating on the same line if the two items include a variable and a "message."

Print Zones



- The TRS-80 is divided into four PRINT zones.
  - — Each PRINT zone has 16 spaces for up to 16 characters.
  - — The TRS-80 can display up to 64 characters per line ($4 \times 16 = 64$).
- Commas are used to tell the computer to move to the next PRINT zone.
  - — The cursor moves to the next PRINT zone each time a comma is encountered.

Print Zones and the Use of Commas

YOUR ACTION	DISPLAY	NOTES
1. Type NEW and press [ENTER].		
2. Type Line 1Ø to read then press [ENTER].	1Ø PRINT "ZONE 1", "ZONE 2", "ZONE 3", "ZONE 4"	
3. Type RUN and press [ENTER].	ZONE 1 ZONE 2 ZONE 3 ZONE 4	(A) (B)
4. Type Line 2Ø to read then press [ENTER].	2Ø PRINT "ZONE 1",, "ZONE 3"	
5. Type RUN and press [ENTER].	ZONE 1 ZONE 2 ZONE 3 ZONE 4 ZONE 1 ZONE 3	(C)

NOTES

(A) There are four (4) 16-character PRINT zones per line (since 4 × 16 = 64, the screen can display up to 64 characters per line).
(B) Note that there are two commas between ZONE 1 and ZONE 3.
(C) The comma tells the computer to move to the next PRINT zone each time a comma is encountered.

Semicolon vs. Comma

YOUR ACTION	THE DISPLAY READS:
1. Type NEW and press [ENTER].	
2. Type exactly as shown then [ENTER]. →	1Ø PRINT "A"; "SEMICOLON"; "PACKS"; "ITEMS"; "CLOSE"; TOGETHER"
3. Type exactly as shown then [ENTER]. →	2Ø PRINT "BUT A", "COMMA", "LEAVES", "SPACES"
4. Type RUN and press [ENTER].	ASEMICOLONPACKSITEMSCLOSETOGETHER BUT A COMMA LEAVES SPACES
5. Type Lines 3Ø, 4Ø, 5Ø, and 6Ø as shown → then [ENTER].	3Ø LET A = 5 4Ø LET B = 1Ø 5Ø LET C = 15 6Ø PRINT A; B; C
6. Type RUN 3Ø and [ENTER].	5 1Ø 15

NOTE

A. There is one exception to this rule for the TRS-80. When the semicolon is used in front of a variable A, B, C, etc., the computer automatically inserts one space. This might not be true with other computers, so beware!

Use of the Semicolon — Summary

- The effect of the semicolon from computer to computer varies, but it is always true that a semicolon leaves less space between the answers or results printed than the COMMA.
- GENERAL RULE: when you want more than one item on the same line and
  - — If you want your results or output spread out, use a comma.
  - — If you want your results or output close together, use a semicolon.
- Exception

  With TRS-80, the semicolon normally tells the computer to print your output close together except when there is a printed message and a variable. For example, in the statement: 6Ø PRINT "THE AREA IS"; A; "SQ. INCHES" the output will look like this (If A = 5Ø):

  The area is 5Ø sq. inches

  Notice that the computer automatically inserted one space between two items when it encountered a semicolon. This is unique to TRS-80.

PRACTICE 6

Area of a Rectangle Program

Part I

1. ENTER and RUN this progam:

```
10 REM**AREA OF A RECTANGLE PROGRAM**
20 REM**AREA (A)=LENGTH(L)*WIDTH(W)**
30 LET L = 10
40 LET W = 5
50 LET A = L*W
60 PRINT A
```

2. Add a new program line to include a label on your answer. For example, the area of the rectangle is 50 square inches.
3. Add new program lines to PRINT the following:
   a. The length of the rectangle is 10 inches.
   b. The width of the rectangle is 5 inches.

Part II

1. *Do not* type NEW.
2. Change the values of L and W in the program. (Think before you change the lines! How many lines do you have to change? Change only those lines!)

PRACTICE 7

Program Using Mathematical Operators

1. ENTER and RUN the following program:

```
10 REM**MATH PROBLEMS**
20 LET A = 75
30 LET B = 50
40 LET C = A+B
50 PRINT C
```

2. Change the values of A and B in the program and RUN it. Fill in the results: A = _____, B = _____, C = _____.
3. Add a program line to label the answer. Example: "The sum is (your answer)."
4. Write a program to multiply (*) two numbers (any two).
5. Add the program line to PRINT: "The product of" (your no.) "*" (your no.) is (your answer). Example: The product of 5 * 5 is 25.
6. Write a program to divide (/) two numbers (any two).
7. Add the program line to PRINT: "The quotient of" (your #) "/" (your #) is (your answer). Example: The quotient of 10/2 is 5.
8. Write a program to subtract (–) two numbers (any two).
9. Add the program line to PRINT: "The difference between" (your #) "–" (your #) is (your answer). Example: The difference between 10–5 is 5.

Additional practices for this Part will be found in the back of the book.

PART 5

Scientific Notation

What You Will Learn

To understand and use scientific notation.

Review and Feedback

The purpose of this part of the program is to evaluate students' overall performance and determine which students are having problems. The students who are having problems will be given the opportunity to review concepts they have not mastered. The review and feedback phase is divided into the following parts:

1. Exam — written/lab
2. Open discussion with students about their concerns and interests
3. Evaluation of student's performance
4. Recommendations

Scientific Notation

- Scientists often express large numbers like 186,ØØØ and small numbers like Ø.ØØØ15 as the product of two numbers. For example:

a) 186,ØØØ = $1.86 \times 1Ø^{5}$
b) Ø.ØØØ15 = $1.5 \times 1Ø^{-4}$
c) 764,ØØØ = $7.64 \times 1Ø^{5}$
d) Ø.Ø347 = $3.47 \times 1Ø^{-2}$
e) 5,ØØØ,ØØØ = $5 \times 1Ø^{6}$

Scientific Notation

Ordinary Notation		Scientific Notation		Scientific Notation in TRS-80	Meaning
5,000,000	=	5×10^{6}	=	5E + 06	ADD 6 zeroes after 5
.000005	=	5×10^{-6}	=	5E − 06	Shift decimal 6 places to left
.00000005	=	5×10^{-8}	=	5E − 08	Shift decimal 8 places left
5 (with 15 zeroes)	=	5×10^{15}	=	5 (with 15 zeroes) NOTE 1	
5 (with 16 zeroes)	=	5×10^{16}	=	5D + 16	ADD 16 zeroes after 5

- The TRS-80 uses scientific notation for very large and very small numbers.
- Rule 1: E + 08 means move the decimal point 8 places to the right.
- Rule 2: E − 08 means move the decimal point 8 places to the left.

NOTE 1: If a constant contains 8 or more digits, that number is stored in double precision and can handle numbers with up to 16 digits (i.e., no conversion to scientific notation is required). If, however, a number has 17 or more digits, that number will be converted to scientific notation. (The "D" is used to denote double precision.)

A Word of Caution about Typing of Constants — (TRS-80)

- Constants are the actual numbers used by Level II BASIC. The following rules determine how the TRS-80 types the number during a program execution.

RULE

1. If a constant contains 8 or more digits, that number is stored in double precision, which means it will print out (type) numbers up to 16 significant digits, and if more than 16 digits convert to double precision scientific notation.

EXAMPLE:	DISPLAY	
10 PRINT	5 000 000 000 000 000, (15 zeroes)	5 000 000 000 000 000 0 (16 zeroes)
RUN	5 000 000 000 000 000 (15 zeroes)	5 D + 16 (converts to double precision scientific notation)

2. If the number is not double precision and if it is outside the range −32768 to + 32767 or if it contains a decimal point, then the number is stored as single precision.

EXAMPLE:	DISPLAY			
10 PRINT	5 000 000,	.000005,	.00000005,	500000
RUN	5E + 06	5E − 06	5E − 08	500000 (Did not convert because within range of single precision significant figures)

3. If neither 1 or 2 is true of the constant, then it is stored as an integer for TRS-80, and integers are whole numbers from − 32768 to + 32767.

Assignment 5-1 — (Scientific Notation)

1. Type, [ENTER], and RUN the following program:

```
 5 CLS
1Ø PRINT 5ØØØ ØØØ, Ø.ØØØØØ5, .ØØØØØØØ5, 5ØØØØØ
15 PRINT
2Ø PRINT 5 ØØØ ØØØ ØØØ ØØØ ØØØ, 5 ØØØ ØØØ ØØØ ØØØ ØØØØ
                 (15 zeroes)              (16 zeroes)
```

2. Do Problems 4-1 and 4-2 (page 23) in TRS-80 Level I Manual.

3. Modify Problem 4-1 as follows:
   a) Change number of cars to 1ØØ million
   b) Change number of miles to 15Ø thousand
   c) Now RUN program

4. Experiment with scientific notation until you feel comfortable with it.

Review and Feedback

A. Quiz — Written/Lab

B. Open discussion with students on concerns and interest

C. Evaluation of student's performance

D. Recommendations

FEEDBACK QUESTIONNAIRE

1. Do you like working with computers? _____ (yes, no) If not, why not? _____
   _____
2. What things do you like most about computers? _____
   _____
3. What do you dislike most about computers? _____
   _____
4. If you were a design engineer and could design the computer to do anything you wanted it to, what kinds of things would you include in your design? (Use your imagination!)
   _____
   _____
5. What was the hardest thing for you to understand about the computer so far? _____
6. What was the easiest thing for you to understand? _____
   _____
7. Were you afraid or nervous when you first used the computer? _____ (yes, no)
8. Do you feel comfortable using the computer now? _____ (yes, no)
9. Would you prefer to be doing something else rather than learning about computers? _____ (yes, no) If yes, what would you like to do? _____
   _____
10. Is the teacher going too fast, too slow, or just right for you? _____
11. Do you find the lessons interesting, boring, or so-so? _____
12. If you could teach this course, what would you do to make the lessons more interesting? _____
    _____
13. Have you decided what you want to do for a vocation? _____ (yes, no)
    If yes, what? _____
14. Would you like to take additional courses to learn more about computers and programming? _____ (yes, no)
15. Do you have any additional comments? _____
    _____
    _____
    _____
    _____
    _____
    _____

PRACTICE 8

Scientific Notation

1. Convert the following to scientific notation (example: 5,000,000 = 5×10^6):
   a. 5,165,123
   b. .000007
   c. .00000008
   d. 6,001,255
   e. 80000 000 000 000 000 (16 zeros)
   f. 8000 000 000 000 000 (15 zeros)
   g. 9,000,156
   h. 7,701,777
   i. 77,701,777
   j. 5612345
2. Change these numbers to scientific notation used in the TRS-80 (example: 5,000,000 = 5E+06).

PART 6

Relational Operators and IF-THEN/GOTO Statements

What You Will Learn

1. How computers compare (or relate) one value with another.
2. To explain the purpose and use of the six relational operators: =, >, <, <=, >=, <>.
3. To explain the purpose and use of the key words IF-THEN, GOTO.
4. To write, enter, and run programs that use IF-THEN and GOTO statements.
5. To understand and use the counting program.

Relational Operators

- Allow computer to compare one value with another.
  - — The three relational operators include

Symbol	Meaning	Examples
=	Equal	A = B
>	Greater than	A > B
<	Less than	A < B

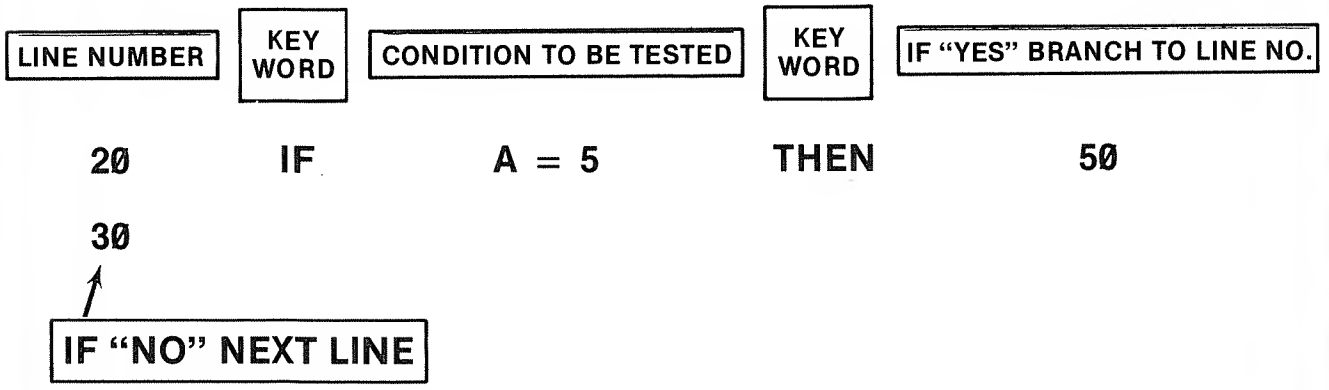
  - — Combining the three operators above we have

<>	Is not equal to	A <> B
<=	Less than or equal to	A <= B
>=	Greater than or equal to	A >= B

NOTE: To distinguish between < and >, just remember that the smaller part of the < symbol points to the smaller of two quantities being compared.

IF-THEN

- IF-THEN is used in conditional branching.
  — That is, the program will "branch" to another part of the program on the condition that it passes the test it contains.
  — If the test fails, the program simply continues to the next line.
- Example:



Sample Program Using IF-THEN (Conditional Branching)

- Program

```
1Ø LET A = 5
2Ø IF A = 5 THEN 5Ø
3Ø PRINT "A DOES NOT EQUAL 5."
4Ø END
5Ø PRINT "A EQUALS 5."
RUN
```

- The screen should display

```
A EQUALS 5
```

- Why is Line 2Ø above a conditional branching statement?
  — What's the condition or test?

In-Class Exercise 6-1 (IF-THEN)

Given: A = 1Ø, B = 2Ø, C = 3Ø

Exercises:

Exercise No.	Statement	Condition is (T or F)	Branch to (Line N) (A)
1.	1Ø IF A = B THEN 4Ø	F	2Ø
2.	1Ø IF A <> B THEN 5Ø		
3.	1Ø IF A > B THEN 6Ø		
4.	1Ø IF A < B THEN 7Ø		
5.	1Ø IF C <= A + B THEN 8Ø		
6.	1Ø IF C >= A + B THEN 9Ø		
7.	1Ø IF B > A THEN 1ØØ		
8.	1Ø IF B/A >= C/A THEN 11Ø		
9.	1Ø IF A * B <= A * C THEN 12Ø		
1Ø.	1Ø IF C/A <= A * B THEN 13Ø		

(A) Note: If condition is false (F), the computer will execute the next line (i.e., 2Ø).

A Counting Program — Using IF-THEN

- Program

```
10 LET J = 0
20 LET J = J + 1
30 PRINT J
40 IF J < 10 THEN 20
RUN
OUTPUT IS*
```

* OUTPUT
1
2
3
4
5
6
7
8
9
10

- In-Class Exercise 6-2

  Modify above program to count to 50 by 5's

IF-THEN Counter Program Analysis

	PROGRAM EXECUTION	"J" COUNTER STATUS	DISPLAY
INITIALIZE	1Ø J = Ø	Ø	
1ST TIME	2Ø J = J + 1	1 = Ø + 1	
	3Ø PRINT J,		
	4Ø IF J < 4 THEN 2Ø		
2ND TIME	2Ø J = J + 1	2 = 1 + 1	
	3Ø PRINT J,		
	4Ø IF J < 4 THEN 2Ø		
3RD TIME	2Ø J = J + 1	3 = 2 + 1	
	3Ø PRINT J,		
4TH TIME	4Ø IF J < 4 THEN 2Ø	4 = 3 + 1	
	2Ø J = J + 1		
	3Ø PRINT J		
	4Ø IF J < 4 THEN 2Ø		
END	5Ø END		1 2 3 4

IF-THEN COUNTER — Program Analysis (Stop-Action)

	PROGRAM EXECUTION	"J" COUNTER STATUS	DISPLAY
INITIALIZE	1Ø J = Ø	1Ø [Ø]	
1ST TIME	2Ø J = J + 1	2Ø [1] = Ø + 1	
	3Ø PRINT J	3Ø	[1]
	4Ø STOP		
	45 REM** TYPE CONT TO CONTINUE**		
	5Ø IF J < 4 THEN 2Ø		
2ND TIME	2Ø J = J + 1	2Ø [2] = 1 + 1	
	3Ø PRINT J	3Ø	[2]
	4Ø STOP		
	45 REM		
	5Ø IF J < 4 THEN 2Ø		
3RD TIME	2Ø J = J + 1	2Ø [3] = 2 + 1	
	3Ø PRINT J	3Ø	[3]
	4Ø STOP		
	45 REM		
	5Ø IF J < 4 THEN 2Ø		
4TH TIME	2Ø J = J + 1	2Ø [4] = 3 + 1	
	3Ø PRINT J	3Ø	[4]
	4Ø STOP		
	45 REM		
	5Ø IF J < 4 THEN 2Ø		
END	6Ø END		

In-Class Exercise 6-3
(GOTO — Unconditional Branching)

- Type and RUN this program:

```
10 CLS
20 PRINT "YOUR NAME";
30 GOTO 20
```

- What happened?
  - — Do you know how to stop the program? (What about the BREAK key!) Explain this simple program (Line 10 merely clears the screen). But what does Line 30 tell the computer to do?
  - — Were there any tests or conditions to be satisfied in Line 30 before it does what it has to do?
  - — Do you understand now why the GOTO statement is called an unconditional branching statement?
- Don't leave this page until you understand everything!

Exercise 6-4 (GOTO/IF-THEN)

Exercise:

- Study the program below and write the message that would be printed if the program were executed.

```
10 PRINT "WELCOME TO LEEDS MIDDLE SCHOOL"
20 GOTO 70
25 PRINT
30 PRINT "HELLO SUPERSTAR"
35 PRINT
40 PRINT "COMPUTERS ARE MY THING"
50 GOTO 100
60 IF A = 5 THEN 90
70 PRINT "COMPUTER WORKSHOP"
80 GOTO 40
90 GOTO 120
100 LET A = 5
110 GOTO 60
120 PRINT "AND I'M A SUPERSTAR!"
130 END
140 PRINT "TRS-80 MICROCOMPUTER"
150 PRINT "I CAN DO IT TOO"
160 PRINT "I SPEAK BASIC"
```

Assignment — 6-1

1) Read Chapter 6 in TRS-80 Level I Manual (page 29).

2) Do Exercises 6-1, 6-2, and 6-3.

3) Write a program of your choice using conditional (IF-THEN) and unconditional (GOTO) statements.

4) Write a counting program.
   — Count to 1ØØ by 1Ø's.

What We Have Learned — Summary

- Relational operators: =, >, <, <>, <=, >=
- IF-THEN
- GOTO (No space between GO and TO)
- Conditional Branching
  - — If condition is met, (i.e., TRUE), branch to designated line in program.
  - — If condition is not met, (i.e., FALSE), go to next line number in program.
- Unconditional branching
  - — GOTO line XX (no conditions or tests required)
  - — A GOTO statement, as the name implies, forces the computer to go to a specific statement anywhere in the program.

PRACTICE 9

Using IF-THEN

Part I.

1. ENTER and RUN the following program:

```
10 LET A = 10
20 IF A = 10 THEN 50
30 PRINT "A DOES NOT EQUAL 10"
40 END
50 PRINT "A EQUALS 10"
```

2. Change Line 10 to Let A = 5 and then RUN it.
3. Change Line 10 to Let A = 3 and then RUN it.

Part II.

1. Using this program as an example, write a new program to PRINT A EQUALS 3 and RUN it.
2. Change the values of A in Line 10 and RUN the program several times.

PRACTICE 10

Counting Program Using IF-THEN

1. ENTER and RUN this program:

```
10 LET J = 0
20 LET J = J+1
30 PRINT J
40 IF J < 10 THEN 20
```

2. Write a program to count from 1 to 15.
3. Write a program to count to 50 by 5's.
4. Write a program to count to 100 by 10's
5. Write a program to count from 15 to 30.
6. PRINT the answers in one column (vertically).
   Example: 15
   16
   17
   18
   and so forth
7. Write a program to count from 20 to 40. PRINT answers horizontally in four columms.
   Example:

20	21	22	23
24	25	26	and so forth

PART 7

Input Statements

What You Will Learn

1. To explain the purpose and use of key words input, input with built-in print.
2. To explain the purpose and use of a trailing semicolon on a program line.
3. To identify and use string variables A$, B$, C$, and so forth.
4. To explain the difference between numeric and string variables.
5. To write, enter, and run programs that use the concepts of this lesson.

Input Statement

STATEMENT	FUNCTION
1Ø INPUT A	• Causes the computer to stop, PRINT a ?, and wait for you to type in a decimal number. • After you type in a value for A, the computer continues the program when you press the ENTER key.

Input Statements

YOUR ACTION	DISPLAY
1. Type NEW and press ENTER.	
2. Type and ENTER Lines 5 & 1Ø as shown.	5 PRINT "THE NUMBER I'M THINKING OF IS" 1Ø INPUT A
3. Type RUN and press ENTER.	THE NUMBER I'M THINKING OF IS ?____ Ⓐ
4. ENTER a number (e.g., type 5 and ENTER).	THE NUMBER I'M THINKING OF IS ? 5
5. RUN this program several times to get the feel of it.	>–

Ⓐ The question mark on the screen means, "It's your turn and I'm waiting."

Input Statements with Built-In Print

YOUR ACTION	DISPLAY
1. Add a semicolon to Line 5 of the resident program (i.e., the program now residing in the computer).	5 PRINT "THE NUMBER I'M THINKING OF IS";
2. RUN the program again.	THE NUMBER I'M THINKING OF IS? _ (A)
3. Change Line 5 to read: →	5 INPUT "THE NUMBER I'M THINKING OF IS"; A
4. Delete Line 10 by typing 10 and then press ENTER.	
5. RUN the program.	THE NUMBER I'M THINKING OF IS? _ (B)

(A) Note that the semicolon puts the question mark on the same line.

(B) The results are exactly the same as before. But here is what was changed:

- — PRINT TO INPUT (Line 5)
- — Eliminated Line 10

Input Statements —
(Area of Rectangle Program)

```
10  REM * AREA OF A RECTANGLE PROBLEM*
20  REM * A = L * W *
30  PRINT "THE LENGTH IS"
40  INPUT L
50  PRINT "THE WIDTH IS"
60  INPUT W
70  A = L * W
80  PRINT "THE AREA IS"
90  PRINT A
```

Area of Rectangle Problem Revisited (Using Input Statements)

YOUR ACTION	DISPLAY
1. Type in program Lines 1Ø through 6Ø as shown.	1Ø REM * AREA OF A RECTANGLE PROBLEM * 2Ø INPUT "THE LENGTH IS"; L 3Ø INPUT "THE WIDTH IS"; W 4Ø A = L * W 5Ø PRINT "THE AREA IS"; 6Ø PRINT A (A)
2. Type RUN then press ENTER. 3. Type in the length (say 1Ø) and ENTER. 4. Type in the width and press ENTER.	THE LENGTH IS?____ THE LENGTH IS? 1Ø THE WIDTH IS?_____ (B)
5. What is your answer?	

(A) Note the trailing semicolon. It is used to hook Lines 5Ø and 6Ø together.

(B) Note that the program waits for an input from the keyboard. If you don't enter a number or press ENTER, it will just stay at that line until the machine is turned off or reset.

Assignment 7-1

1. Read Chapter 7 (page 33) in TRS-80 Level I Manual.
2. Write a simple program to do the following:
   (using input statement)
   a) Input your age
   b) Input your zip code
   c) Input your weight
   d) Input your height in inches
   e) PRINT each of the above with the proper labels
      (for example: My age is 15 or I am 15 years old).

What We Have Learned

- Trailing semicolon hooks two lines together.
- Input statements cause the computer to stop and wait for an input from the keyboard.
- Input statements can have a built-in message to tell you what to input.

Numeric vs. String Variables

Numeric Variable		Declaration Character'(1)		String Variable
A	+	$	=	A$
A1	+	$	=	A1$
AB	+	$	=	AB$
AZ	+	$	=	AZ$

(1) NOTE: Simply by adding the string declaration character ($) to the numeric variable allows you to use any numeric variable as a string variable.

Example of Use of String Variables

YOUR ACTION	DISPLAY
1. Type and ENTER.	10 CLS 20 INPUT "YOUR NAME IS"; A$ 30 PRINT "HELLO THERE,"; A$
2. RUN and ENTER.	YOUR NAME IS? HELLO THERE, BILL Ⓐ

Ⓐ **NOTE:**
It will print your name and not "BILL," unless your name is "BILL."

In-Class Exercise 7-1 (String Variables)

YOUR ACTION	DISPLAY
Type and ENTER. →	5 CLS 1Ø INPUT "YOUR FIRST NAME" ; A$ 2Ø INPUT "YOUR MIDDLE NAME" ; B$ 3Ø INPUT "YOUR LAST NAME"; C$ 4Ø PRINT A$; " "; B$; " "; C$ 5Ø INPUT "YOUR FULL NAME"; D$ 6Ø PRINT D$
RUN and ENTER. →	YOUR FIRST NAME? AUBREY YOUR MIDDLE NAME? BRIGHT YOUR LAST NAME? JONES AUBREY BRIGHT JONES YOUR FULL NAME? AUBREY BRIGHT JONES AUBREY BRIGHT JONES

(Sample)

Ⓐ

Ⓐ NOTES

You can add string variables together.

You must insert a space between string variables using " " marks.

A semicolon will not cause a space to be printed.

Assignment 7-2 (String Variables)

1. RUN and analyze the following program:

```
Program A
 1Ø  INPUT "YOUR NAME IS" ; A$
 2Ø  INPUT "YOUR HOUSE NUMBER" ; A
 3Ø  INPUT "YOUR STREET NAME" ; B$
 4Ø  INPUT "YOUR ZIP CODE" ; B
 5Ø  PRINT A$
 6Ø  PRINT A; "  " ; B$
 7Ø  PRINT "ZIP CODE" ; B
```

2. Answer the following questions:
   a) Why were A$ and B$ (string variables) required in Lines 1Ø and 3Ø?
   b) Why were quotes (" ") inserted in Line 6Ø?
   c) Why did we use $ symbol (or string declaration character) with A and B in Lines 2Ø and 4Ø?

String Variables — Summary

- String variables can be assigned to indicate letters, words, and/or combinations of letters.
- Although it is possible to string up to 255 characters per string variable, you should limit your strings to 5Ø characters per string variable for now. (We'll show you how to get 255 later!)
- String variables can be added together.
- Use " " marks to insert a space between string variables.

PRACTICE 11

Area of Rectangle Problem (Using INPUT Statement)

1. ENTER and RUN this program:

```
10 REM*AREA OF RECTANGLE PROBLEM*
20 INPUT "THE LENGTH IS"; L
30 INPUT "THE WIDTH IS"; W
40 LET A = L*W
50 PRINT "THE AREA IS"; A
```

2. Write a new program using INPUT statements to find volume (volume = length × width × height).
3. Include a statement: The volume is _____.

PRACTICE 12

More INPUT Statement Programs

Part I.

1. Write a program using INPUT statements to change meters to centimeters (centimeters = 100 × meters).
2. Include a statement: _____ meters equals _____centimeters.

Part II.

1. Write a new program using INPUT statements to do the following:
   a. Input your age.
   b. Input your zip code.
   c. Input your weight.
   d. Input your height.
2. PRINT each with the proper labels.
   Example: My age is _____.

PRACTICE 13

String Variables

Part I.

1. ENTER and RUN the following program:

```
10 INPUT "YOUR NAME IS"; A$
20 INPUT "YOUR HOUSE NUMBER"; A
30 INPUT "YOUR STREET NAME"; B$
40 INPUT "YOUR ZIP CODE"; B
50 PRINT A$
60 PRINT A; "  "; B$
70 PRINT "ZIP CODE"; B
```

2. Answer the following questions:
   a. Why are A$ and B$ (string variables) required in Lines 10 and 30?
   b. Why were quotes (" ") inserted in Line 60?
   c. Why didn't we use $ symbol (or string declaration character) with A and B in Lines 20 and 40?

Part II.

1. Write a new program using INPUT statements, string variables, and a space between each line. PRINT all information (example: My best friend is _____) to give the following information:
   a. Your best friend.
   b. Your favorite subject.
   c. Your favorite food.
   d. Your favorite movie star.
   e. Your favorite color.
   f. Your zodiac sign.

PART 8

Using the Calculator Mode and Sizing Memory

What You Will Learn

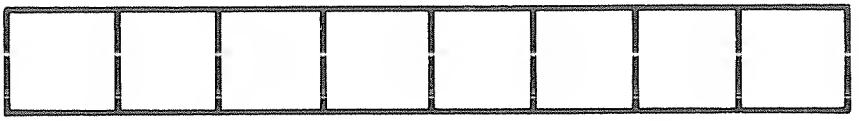
1. To define and use the terms bit, byte, k, kbytes.
2. To determine how much memory is used in a BASIC program.
3. To explain the purpose and use of the command PRINT MEM.
4. To use the TRS-80 in calculator mode (i.e., without having to write a program).

BIT = BINARY—DIGIT

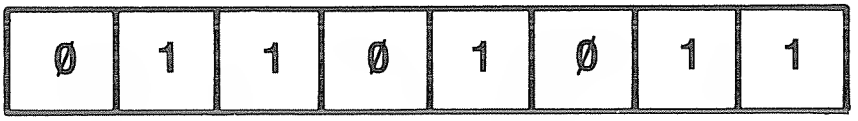
BIT = SMALLEST MEMORY CELL IN A COMPUTER

BIT = "1" OR "Ø"

MEMORY CELL WITH 1 BIT



8 MEMORY CELLS



8 BITS = 1 BYTE

BYTE = 8 BITS

K = 1ØØØ

KBYTES = 1ØØØ BYTES

KBYTES = 8ØØØ BITS

How Much Memory Is Used in BASIC Programs

WHAT'S STORED	HOW MUCH MEMORY
1 ALPHA CHARACTER (A-Z)	1 BYTE
1 SPECIAL CHARACTER (e.g., ", !, +, −, etc.)	1 BYTE
1 NUMERIC CHARACTER (Ø-9)	1 BYTE
1 SPACE	1 BYTE
1 LINE NUMBER	3 BYTES } MEMORY
1 ENTER KEY	1 BYTE } OVERHEAD*

EXAMPLE:

10 PRINT "MY NAME IS AUBREY" ENTER

3+1 + 5 + 1 19 1 = 30 BYTES

* Memory overhead means you will use 4 bytes of memory for each line, short or long.

The Memory Command

- PRINT MEM
  - — This command is used to let you know how much memory is available to you.
  - — Sometimes it may be important to know how much memory you are using for a given program.
  - — The amount of memory available in the TRS-80 you are using is 16k. This means that there are about 16,ØØØ different memory locations to store and process your programs (actually 16,384).
- Note!
  - — With no program loaded, there are 15,572 memory locations available for use. The difference in memory space between 15,572 and 16,384 is set aside for processing programs and overall management and monitoring of what the computer is doing.

Assignment 8-1

1. Determining available memory:
   a) Type NEW and press [ENTER].
   b) Type PRINT MEM and press [ENTER].
   c) Display reads: 15, 572.
   d) Now type the following and [ENTER]
      1Ø PRINT "LEEDS MIDDLE SCHOOL."
   e) Type PRINT MEM and press [ENTER].
   f) How much space is left in memory? _____

2. Read Chapter 8 in TRS-80 Level I Manual (page 37).

3. Use TRS-80 in calculator mode to solve the following:
   a) 25 * 4/2
   b) (25 + 6) − 7 + (2 * 5)
   c) 7/2 * 5 * 2 ↑ 3
   d) Any other problems you want to try

Remember! You don't need a line number for calculator mode. Simply type PRINT and the calculations you want done. Example: If you wish to multiply 2 asterisk 3, simply type PRINT 2 * 3, and press [ENTER]. The answer (6) will be displayed.

What We Have Learned

- COMPUTERS SPEAK IN MACHINE LANGUAGE
- MACHINE LANGUAGE IS A FORM OF BINARY CODING
- BINARY CODE CAN BE EITHER "Ø" OR "1" BITS
- BIT = BINARY DIGIT
- BYTE = 8 BITS
- YOU DO NOT HAVE TO KNOW MACHINE LANGUAGE TO USE COMPUTERS!

PRACTICE 14

Sizing Memory and Calculator Mode

Part I.

1. To determine available memory:
   a. Type NEW and press [ENTER] .
   b. Type PRINT MEM and press [ENTER] .
   c. Display reads: 15,572.
   d. Now type the following and [ENTER] 10 PRINT "LEEDS MIDDLE SCHOOL."
   e. Type PRINT MEM and press [ENTER] .
   f. How much space is left in memory? _____

Part II.

1. Use TRS-80 in calculator mode to solve the following:
   a. 25 * 4/2
   b. (25 + 6) − 7 + (2 * 5)
   c. 7/2 * 5 * 2 ↑ 3
   d. Any other problems you want to try.

PART 9

Using the Cassette Recorder

What You Will Learn

1. How to use the cassette as an output device to save information stored in memory.
2. How to use the cassette as an input device to load information from tape to memory.
3. To explain and use the commands CSAVE, CLOAD.
4. To make critical settings on the tape recorder and to practice using the recorder.

A Cassette Recorder Is an I/O Device

Using Cassette Tape Recorder

- The cassette tape recorder is an input/output (I/O) device that allows you to "save" information on cassette or "load" information from cassette.
  - — When you have typed a long program and wish to save it, you can save it on (CSAVE) cassette.
  - — When you are ready to use it again, you can load (CLOAD) it from the cassette.
- Note! You can only save your program on cassette (not the program output).
- Refer to the TRS-80 Level II Reference Manual for tips on using the recorder.

Using the Tape Cassette Recorder as an Output Device (That Is, to Save a Program from Memory)

STEP	ACTION
1.	Place blank tape in recorder.
2.	Wind tape past leader (the non-magnetic part of the tape).
3.	Set volume at appropriate level.
4.	For Level II TRS-80's: Set volume at 4–6 for CTR-41, 3–5 for CTR-80.
5.	(For Level II) Type CSAVE "PROGRAM NAME" When program name can be any variable (for example, CSAVE "A", CSAVE "B", etc.)
6.	Press PLAY and RECORD button on recorder.
7.	Press ENTER.
8.	When program is saved, recorder will stop and "READY" will appear on screen.

Using the Tape Cassette Recorder as an Input Device (That Is, to Load a Program from Tape Into Memory)

STEP	ACTION
1.	Place program cassette in recorder.
2.	Rewind tape (if necessary).
3.	Set volume at appropriate level (note: same volume used to save tape).
4.	Type CLOAD or CLOAD "A" if program was saved with CSAVE "A".
5.	Two asterisks will appear on the screen and one will flash to let you know the block of data is being transferred from tape recorder to memory.
6.	If one asterisk does not flash or neither asterisk appears, try adjusting the volume control and try again. (Sometimes this can be a very frustrating experience, but don't give up!)
7.	When program is loaded, "READY" will appear on screen.

PRACTICE 15

Using the Computer to Solve Problems

1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer (output).
   The total enrollment at Armstrong High School is 1,264. There are 367 freshmen, 322 sophomores, and 298 juniors. How many seniors are there?
2. Write a new program using INPUT statements to solve one of the problems.

PRACTICE 16

Finding the Average Problems

1. Write a program to solve the following problem. Include a PRINT statement in your program to describe your answer.
   The weights of three boys are 140 lb, 150 lb, and 130 lb. What is their average weight?
2. Write a new program using INPUT statements to solve the same problem. (That is, you should use the INPUT statement for the weight of the three boys.)

PRACTICE 17

Using the Computer to Solve Problems

1. Write two programs to solve the following problems. Label your answers.
2. Over a period of six years Mr. Smith drove his car 53,862 miles. What was the average distance each year?
3. After 12 dozen bulbs were sold, how many of the 1,000 bulbs were left?

PART 10

Using FOR-NEXT-STEP Statements

What You Will Learn

1. To explain the purpose and use of key words FOR-NEXT-STEP.
2. To explain the purpose and use of the terms increment, decrement, initialize.
3. To compare key words GOTO, IF-THEN, FOR-NEXT and explain how they relate to one another.
4. To explain the purpose and use of timer loops.

For-Next Statement

- Allows the computer to do the same thing over and over a large number of times (and do it very fast!)

FOR - NEXT Loop

YOUR ACTION	DISPLAY
1. Type and ENTER program as shown. →	5 CLS 10 FOR J = 1 TO 10 20 PRINT " AUBREY" ; J 30 NEXT J
2. Type RUN and ENTER.	AUBREY 1 AUBREY 2 AUBREY 3 AUBREY 4 AUBREY 5 AUBREY 6 AUBREY 7 AUBREY 8 AUBREY 9 AUBREY 10

FOR-NEXT-STEP Loop

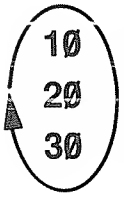
YOUR ACTION	DISPLAY
1. Retype and ENTER Line 1Ø of resident*program as shown. →	1Ø FOR J = 1 TO 1Ø STEP 3 Ⓐ
2. Type RUN and ENTER.	AUBREY 1 AUBREY 4 AUBREY 7 AUBREY 1Ø

*Resident means program currently in memory.

Ⓐ If step is not included in the statement, an increment of 1 is assigned by the computer (i.e., step 1).

Example of Program Statements
Using Key Words

FOR-NEXT-STEP



```
10 FOR J = 10 TO 1 STEP -1
20 PRINT J ;
30 NEXT J
```

```
RUN
DISPLAY READS:
10  9  8  7  6  5  4  3  2  1
```

Analysis of FOR-NEXT-STEP Statements

LINE NO.	KEY WORD	COUNTER VARIABLE		INITIAL VALUE		FINAL VALUE	INCREMENT/ DECREMENT
1Ø	FOR	J	=	1Ø	TO	1	STEP -1
2Ø	PRINT	J					
3Ø	NEXT	J					

The FOR-NEXT-STEP loop works as follows: The first time the FOR statement is executed, the counter is set for the initial value "1Ø." Then it executes Line 2Ø (PRINT J). When the program reaches Line 3Ø (NEXT J), the counter is decremented by the amount specified (Step-1). If this step has a positive value, the counter is incremented by the amount specified (e.g., Step 2 means increment by 2's).

Comparison of GOTO, IF-THEN, and FOR-NEXT Program Loops

A.

GOTO

(Unconditional Loop)

```
 5 CLS
10 PRINT "AUBREY"
20 GOTO 10
RUN
```

- Program loops one zillion times!
  (or until you stop it)

B.

IF-THEN

(Conditional Loop)

```
 5 CLS
10 LET J = 0
20 J = J + 1
30 IF J>6 THEN 99
40 PRINT "AUBREY" ; J
50 GOTO 20
99 END
RUN
```

- This program loops 6 times!

C.

FOR-NEXT

(Conditional Loop)

```
 5 CLS
10 FOR J = 1 TO 6
20 PRINT = "AUBREY" ; J
30 NEXT J
99 END
RUN
```

- This program loops 6 times!

Comparison of GOTO, IF-THEN, and FOR-NEXT Program Loops

A. "DUMB LOOP"

```
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
AUBREY
```

B. "SMART LOOP"

```
AUBREY 1
AUBREY 2
AUBREY 3
AUBREY 4
AUBREY 5
AUBREY 6
```

C. "SMART LOOP"

```
AUBREY 1
AUBREY 2
AUBREY 3
AUBREY 4
AUBREY 5
AUBREY 6
```

NOTE: Press BREAK Key to Get Out of Loop.

FOR-NEXT SUMMARY

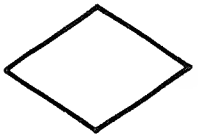
- FOR– NEXT – STEP

— FOR – NEXT is always used as a pair.
— If the key word "step" is not used, the increment of 1 is assumed.
— If the step has a negative value, the counter is decremented (e.g., for J = 1Ø to 1 step -1).
— If the step has a positive value, the counter is incremented (e.g., for J = 4 to 1Ø step 2).

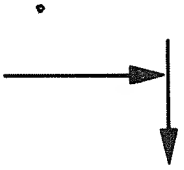
Flowchart Symbols

- Begin or End

- Processing Block



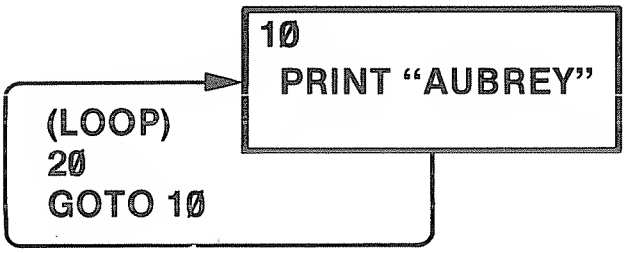
- Decision Diamond



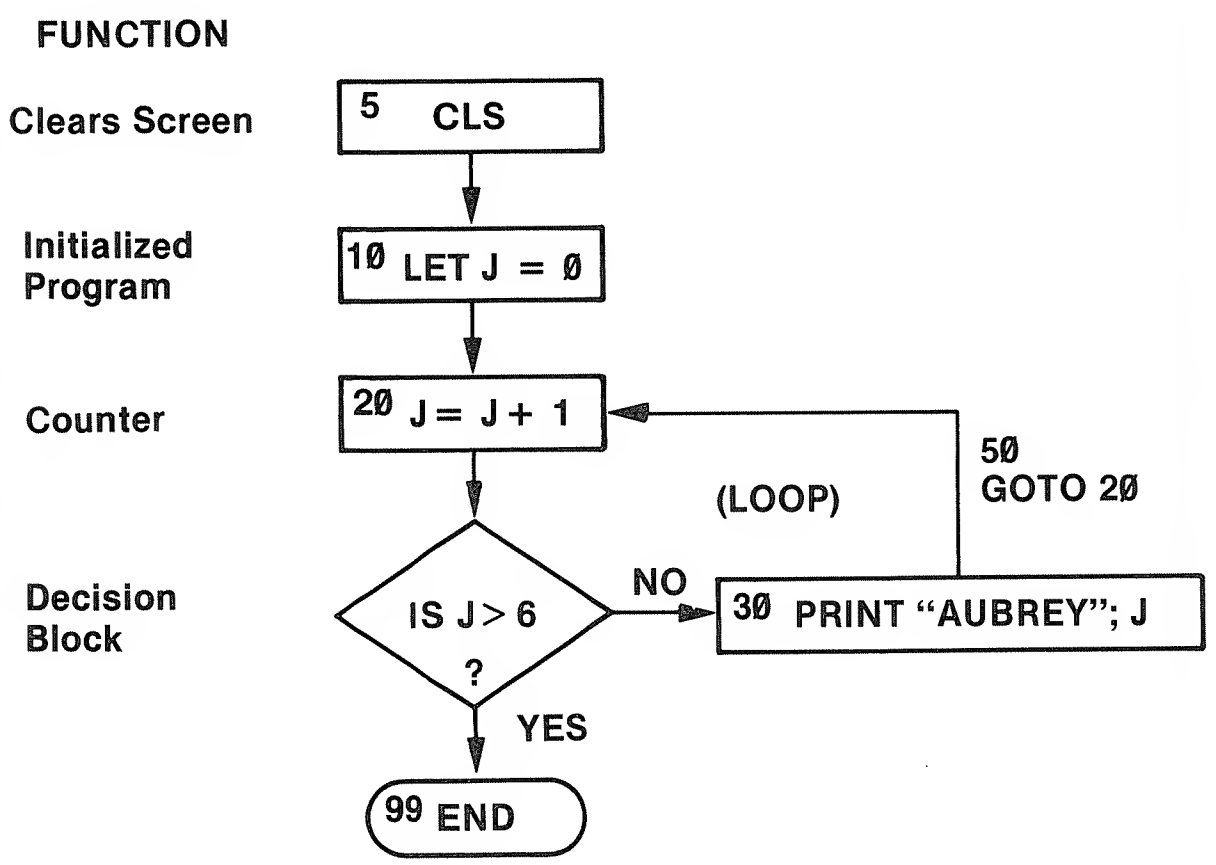
- Connector Arrows

GOTO-LOOP

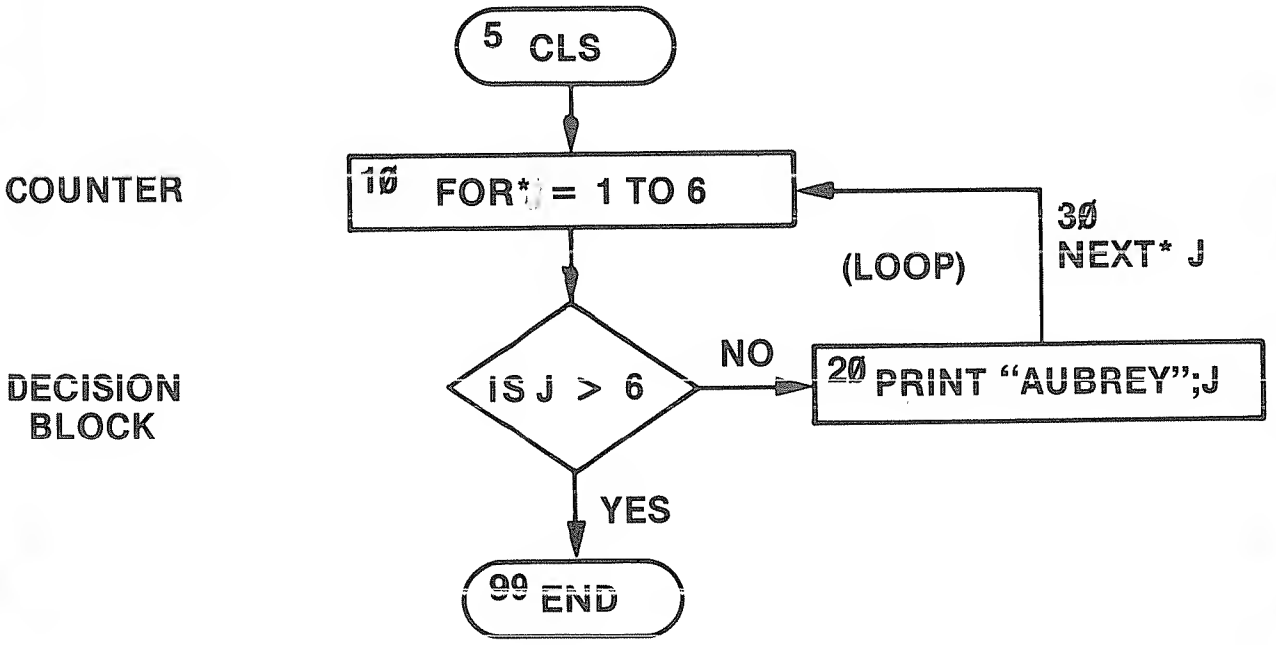
(Unconditional)



Looping with IF-THEN



Looping with FOR-NEXT



*FOR-NEXT *Work together as a counter

Timer Loop

- The TRS-80 can do approximately 5ØØ FOR-NEXT loops per second.
- Example

```
 5 REM* 1Ø SECOND TIMER PROGRAM *
1Ø PRINT "TIMER PROGRAM COUNTING"
2Ø FOR X = 1 TO 5ØØØ
3Ø NEXT X
4Ø PRINT "TIMER PROGRAM ENDED"
```

- You don't believe the TRS-80 can count? Well, try it! (Type in the above program and RUN.)
  — Don't forget to use your watch!

Assignment 10-1

1. Read Chapters 1Ø, 11, and 13 in TRS-80 Level I Manual (chapters start on pages 45, 53, and 65, respectively).
2. Do all the exercises in Chapter 1Ø.
3. Do Exercise 11-1 (page 54).
4. Do Exercise 11-2 (page 57).
5. Do Exercise 13-1 and 13-2 (page 68).

PRACTICE 18

Counting Programs Using IF-THEN and FOR-NEXT

1. Using IF-THEN, write a program to count 5's from 50 to 5.
   a. Written vertically
   b. Written horizontally
2. *Do not* type NEW (that is, save the program above).
3. Using FOR-NEXT, write a program to count to 50 by 5's written horizontally.
   Note: Start your second program at Line 100. That is, type Line 100 as follows: 100 PRINT : PRINT (Of course, this is to insert two spaces between your outputs.)
4. How many program lines (excluding Line 100) did it take using FOR-NEXT? _____
   How many using IF-THEN? _____
5. What can you conclude from this task?

PRACTICE 19

Using IF-THEN and FOR-NEXT Statements

1. Using IF-THEN, write a program to generate all the even numbers between 11 and 51 from smallest to the largest (that is, 12, 14, 16, and so forth).
2. *Do not* type NEW.
3. Using FOR-NEXT, write a program that generates the same numbers and PRINT them horizontally. (*Note:* Start at Line 100. Type Line 100 as → 100 PRINT : PRINT and your next line should be 110.)
4. Type NEW and ENTER .
5. Using IF-THEN, write a program to generate all even numbers between 11 and 51 from largest to the smallest.
6. Do the same using FOR-NEXT.

PART 11

Reading Data

What You Will Learn

1. To explain the purpose and use of the key words READ, DATA, RESTORE.
2. To compare the three different ways you have learned to input data into the TRS-80.
3. To write, enter, and run programs using READ-DATA and READ-RESTORE key words.

READ-DATA

READ-DATA statements are much more efficient than INPUT or LET statements when you have lots of data to input.

Ways of Inputting Data to the Computer (i.e., Ways We've Learned So Far)

10 LET A = 5	10 INPUT A	10 DATA (5) 20 READ A
BUILT-IN	FROM KEYBOARD	READ-DATA COMBINATION

Ways of Inputting Data to the Computer

STATEMENT	FUNCTION
• 1Ø LET A = 5	• LET statement builds value into the program.
OR	
• 1Ø INPUT A	• INPUT statement allows you to [ENTER] data through the keyboard.
OR	
• 1Ø DATA (5)	• DATA statement contains the value (5), which will be stored in a specified variable.
2Ø READ A	• READ statement names the variables in which the values are to be stored.

NOTES: Data lines can be read only by READ statements.
The READ-DATA work together to input data to the computer.

READ-DATA Example

```
5 REM*READ — DATA EXAMPLE*
10 DATA 1, 2, 3, 4, 5
20 READ A, B, C, D, E
30 PRINT A, B, C, D, E
```

DATA STATEMENT → 10 DATA

READ STATEMENT → 20 READ

PRINT STATEMENT → 30 PRINT

NOTES:

- Each piece of data must be read by a READ statement.
- Each READ statement can read a number of pieces of data if each variable is separated by a comma.
- Data lines can only be used by READ statements.

Exercise 11-1 (Reading Data)

Type and ENTER.

```
1Ø DATA 1, 2, 3, 4, 5
2Ø READ A, B, C, D, E
3Ø PRINT A, B, C, D, E
```

Type RUN and ENTER.

```
 1   2   3   4
 5
```

NOTES:

- **The display shows that all five pieces of data in Line 1Ø were read by Line 2Ø, assigned letters A through E, and printed by Line 3Ø.**
- **Data lines are always read left to right by READ statements.**

READ-DATA Summary (Key Words)

DATA

- Key word that lets you store data inside your program to be accessed (read) by READ statements.
  - — Data items will be read sequentially starting with the first item in the first DATA statement and ending with the last item in the last DATA statement.
  - — Items in data list may be string or numeric constants.
  - — If string values include leading blanks, colons, or commas, you must enclose these values in quotes.
  - — DATA statements must match up with the variable types in the corresponding READ statement.
  - — DATA statements may appear anywhere it is convenient in a program.

- EXAMPLE:

```
10 DATA "JONES, A.B.","SMITH, R.J."
20 DATA LEEDS MIDDLE SCHOOL, COMPUTERS
30 DATA 125, 250, 750, 1000
```

NOTE: Quotes used here because of commas

READ-DATA Summary (Key Words)

READ

- Key word that instructs the computer to read a value from a DATA statement and assign that value to the specified variable.
  - — The first time a READ statement is executed, the first value in the first DATA statement is used; the second time, the second value in the DATA statement is used. When all the items in the first DATA statement are used (READ), the next READ will use the first value in the second DATA statement, and so on.
  - — An out-of-data error (OD) occurs if there are more attempts to READ than there are data items.

- EXAMPLE:

  40 READ A$, B$, C$, D$, A, B, C, D

  (Note that there are eight READ variables and eight DATA items on previous page for program Lines 10, 20, and 30)

Assignment 11-1

1. Type and ENTER the following program:

```
1Ø PRINT "NAME", "GRADE"
2Ø READ A$
3Ø IF A$ = "END" THEN PRINT "END OF LIST": END
4Ø READ G
5Ø IF G < 75 THEN PRINT A$, G
6Ø GOTO 2Ø
7Ø DATA "GRAY, BILL", 95, "JONES, A.B.", 65
8Ø DATA "JONES, A.C.", 1ØØ, "SMITH, R.L.", 7Ø
9Ø DATA "EPPS, S.W.", 6Ø, "WELLS, DAVE", 1ØØ, END
```

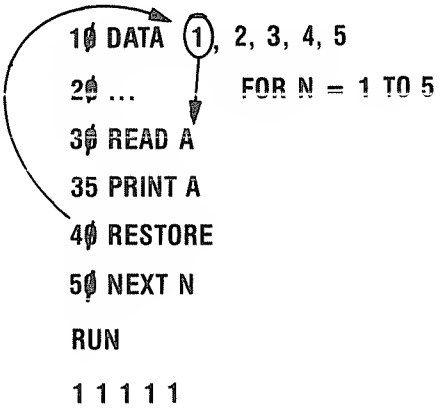
2. Predict the output of the program.
3. Why were quotes used in the DATA statements?
4. RUN the program and record the results.

RESTORE

- Key word that causes the next READ statement executed to start over with the first DATA statement.
  - — This lets your program reuse the same data lines.
  - — Sometimes it is necessary to READ the same data more than once without having to run the complete program again; therefore, RESTORE is used.
  - — Whenever the program comes to RESTORE, all data lines are restored to their original unread condition, both those lines that have been READ and those that have not been READ. This allows all data to be available for reading again, starting with the first data item in the first data line.

NOTE! Remember that each piece of data in a data line can only be read once each time the program is RUN. The next time a READ statement requests a piece of data, it will READ the next piece of data in the data line, or, if data on that line are all used up, it will go to the next data line and start reading it. Therefore, the RESTORE statement is needed if the same data is to be READ more than once in the same program.

Illustration of the READ-RESTORE Feature



NOTE:

- RESTORE caused data Line 10 to be restored to its original unread condition, making all data available for reading again.
- Since there is only one read variable, A, it starts with the first piece of data, 1, in this case.

Exercise 11-2 (READ-RESTORE Data in a FOR-NEXT Loop)

YOUR ACTION	DISPLAY	
1. Type and ENTER.	10 DATA 1, 2, 3, 4, 5 20 FOR N = 1 TO 5 30 READ A 40 PRINT A ; 50 NEXT N	
2. Type RUN and ENTER.	1 2 3 4 5	
3. Insert Line 35. (Type and ENTER)	35 RESTORE	Restores Data Line to Its Original Unread Condition
4. Type RUN and ENTER.	1 1 1 1 1	Therefore Computer Reads First Data Item Over and Over

READ-DATA SUMMARY

- READ-DATA
  - — Key words used to input lots of data to the computer.
- RESTORE
  - — Key word used to restore (put back) data so it can be used again.
- Data lines can be read only by READ statements.
  - — If more than one piece of data is placed on a data line, they must be separated by commas. Each piece of data must be read by a READ statement.
- Data lines are read from left to right by READ statements.
  - — Data lines can be placed anywhere in a program.
- READ-DATA statements are extremely common.
  - — RESTORE is used less often.

PRACTICE 20

READ-DATA

1. Type and [ENTER] the following program:

```
 5 CLS
10 PRINT "NAME", "GRADE"
20 READ A$
30 IF A$ = "END" THEN PRINT "END OF LIST":END
40 READ G
50 IF G > 75 PRINT A$, G
60 GOTO 20
70 DATA "GRAY,BILL", 95,"JONES, A.B.", 65
80 DATA "JONES,A.C.", 100, "SMITH, R.L.", 70
90 DATA "EPPS, S.W.", 60, "WELLS, DAVE", 100, END
```

2. Predict the output of the program.
3. Why were quotes used in the data statements?
4. RUN the program and record the results.

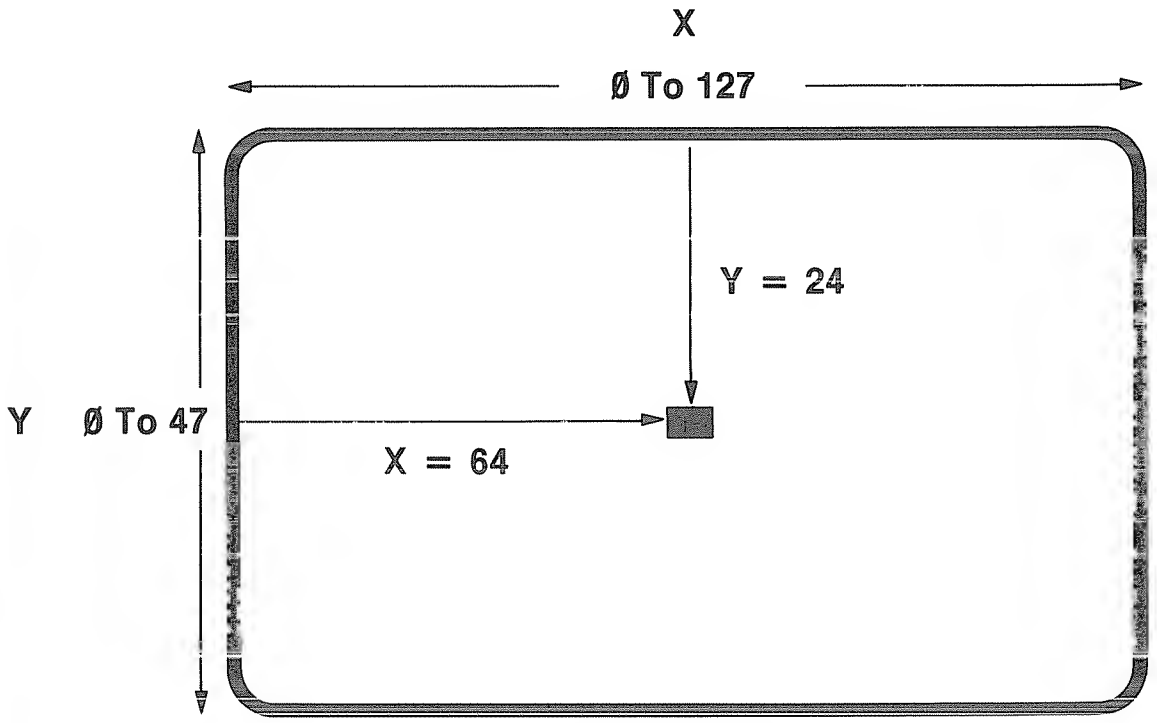
PART 12

Video Display Graphics

What You Will Learn

1. To explain the purpose and use of key words SET, RESET, TAB, PRINT AT, and POINT.
2. To become familiar with the layout of the TRS-80 display via the video display worksheet.
3. To draw pictures and letters on the screen.
4. To write, enter, and run programs utilizing all the concepts learned in this lesson.

Video Display Layout Showing X, Y Coordinates Used with SET/RESET Commands



SET (64, 24)
RESET (64, 24)

Graphic Commands — SET/RESET

Key Word	Function	Example
• SET	• Turns on (or lights up) a particular "light" on the screen	• 1Ø SET (64, 24)
• RESET	• Turns off (or blackens) a particular "light"	• 2Ø RESET (64, 24)

Note:

SET (55,32) means "turn on the light" at the junction where X = 55 and Y = 32. X is the horizontal address counting across from the left-hand side of the screen. Y is the vertical address counting down from the top of the screen. Therefore, everything starts from the upper left-hand corner.

In-Class Exercise 12-1 (SET/RESET)
Use Your Video Display Worksheet for the Following

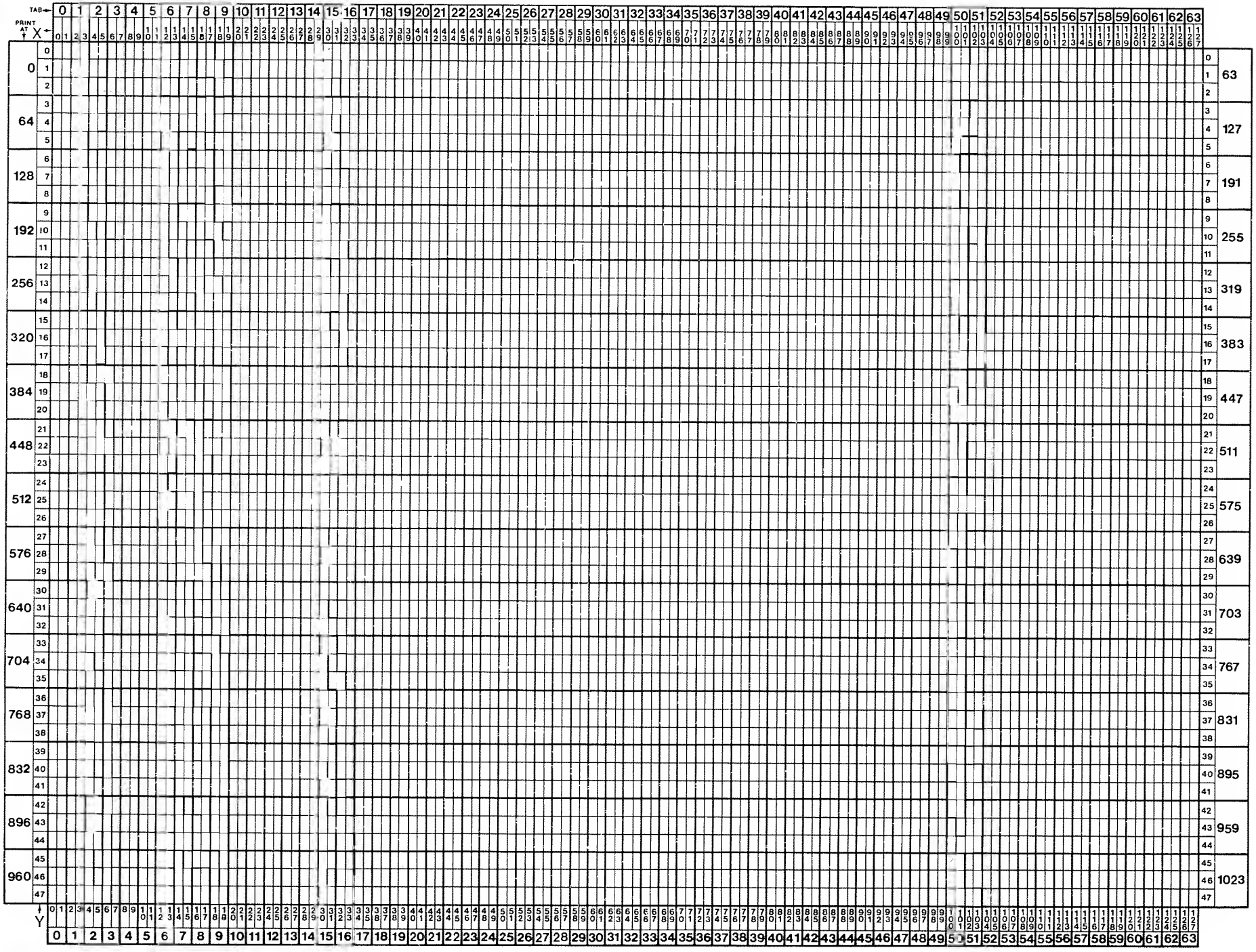
1. Locate X at the top left of the worksheet.
   The X numbers go from Ø to _____.
2. Locate Y at the bottom left of the worksheet.
   The Y numbers go from Ø to _____.
3. Locate the following points on your worksheet:
   a) SET (55,35)
   b) SET (1ØØ, 45)
   c) SET (15, 15)
   d) SET (2Ø, 25)
4. Study the following programs and predict the output:

a)
```
1Ø CLS
2Ø X = 6Ø
3Ø Y = 15
4Ø SET (X, Y)
5Ø RESET (X, Y)
6Ø GOTO 4Ø
```

b)
```
1Ø CLS
2Ø FOR X = Ø TO 127
3Ø FOR Y = Ø TO 47
4Ø SET (X, Y)
5Ø NEXT Y
6Ø NEXT X
99 GOTO 99
```

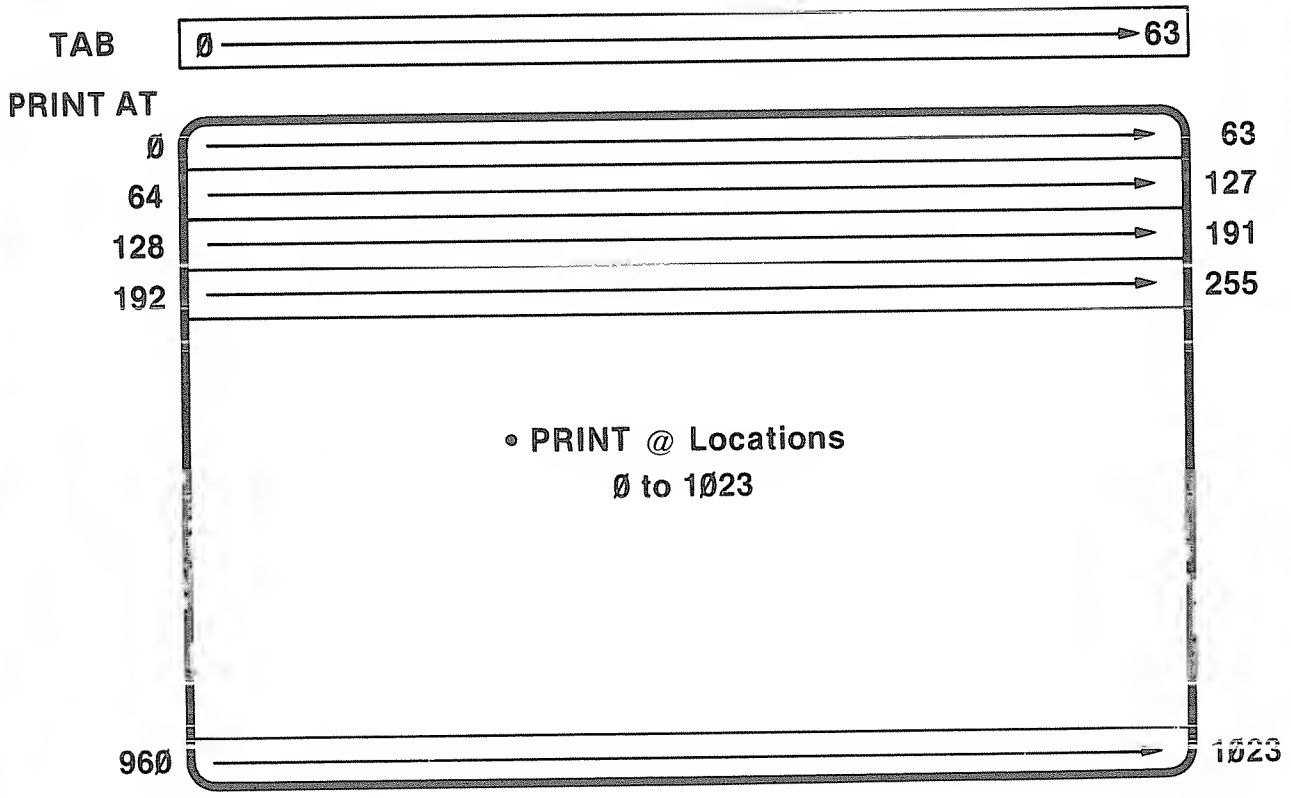
TRS-80 Video Display Worksheet

TITLE _____ PROGRAMMER _____ COMMENTS _____ PAGE ____ OF ____



Courtesy of Radio Shack, a Division of Tandy Corporation.

TRS – 80 Video Display



- **PRINT TAB (Locations: Ø To 63)**
  Example:
  — 1Ø PRINT TAB (2Ø) "TABBED 2Ø"

- **PRINT @ (Locations: Ø To 1Ø23)**
  Example:
  — 1Ø PRINT @ 5ØØ, "HELLO 5ØØ"

In-Class Exercise 12-2 (PRINT @ and TAB)

1. Using your video display worksheet, locate the following:

a) PRINT @ 550	e) PRINT @ 63	i) TAB (15)
b) PRINT @ 1000	f) PRINT @ 960	j) TAB (30)
c) PRINT @ 165	g) PRINT @ 1023	k) TAB (63)
d) PRINT @ 0	h) PRINT @ 350	l) TAB (33)

2. Using the TRS-80, type, ENTER, and RUN the following:

   a) 100 PRINT @ 550, "LOCATION 550"

   b) 100 PRINT @ 1000, "LOCATION 1000"; (Note the trailing semicolon)

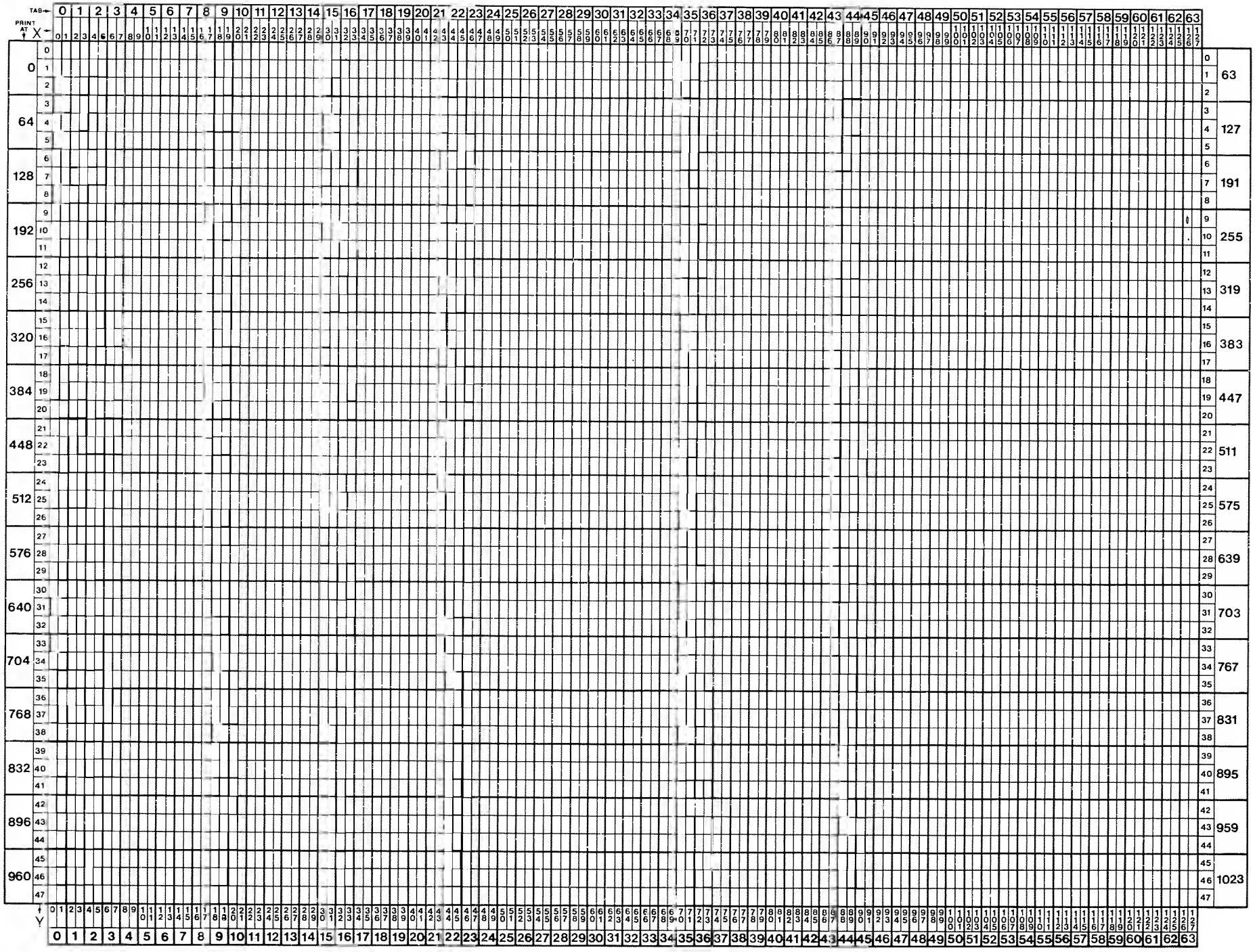
   c) 10 CLS
      20 PRINT @ 200, "HELLO THERE 200, WHEREVER YOU ARE."

   d) NEW
      10 PRINT TAB (5) "TABBED 5"; TAB(25) "TABBED 25"

   e) NEW
      5 LET X = 3
      10 PRINT TAB (X) X; TAB (X↑2) X↑2; TAB (X↑3) X↑3

TRS-80 Video Display Worksheet

TITLE _____ PROGRAMMER _____ COMMENTS _____ PAGE ____ OF ____



TAB and PRINT @ Examples

YOUR ACTION	DISPLAY
1. Type and ENTER. →	1Ø PRINT TAB (2Ø) "YOUR NAME" 2Ø PRINT 3Ø PRINT TAB (2Ø) "YOUR ADDRESS" 4Ø PRINT 5Ø PRINT TAB (2Ø) "YOUR TELEPHONE NO."
2. Type RUN and ENTER.	AUBREY JONES 914 E. SEDGWICK ST. 123-4567
3. Type and ENTER. →	1ØØ PRINT @ 55Ø, "LOCATION 55Ø"
4. Type RUN and ENTER.	LOCATION 55Ø

Summary and Assignment — SET/RESET

- Summary
  - — SET and RESET work together to act as an on/off switch for lights on the display.
  - — The X coordinate goes from Ø to 127, and the Y coordinate goes from Ø to 47.
- Assignment 12-1
  - — Read Chapter 12, pages 61 through 63, in the TRS-80 Level 1 BASIC Manual and do Exercise 12-1.
  - — Study video display worksheet. Note the numbers next to tab go from Ø to 63.

PRINT @ (Print At) — Summary

- A special type of PRINT statement particularly useful in graphics.
  - — Specifies exactly where printing is to begin.
  - — The location specified must be a number from Ø to 1Ø23.
  - — Example:

    ```
    1ØØ PRINT @ 55Ø, "Location 550"
    ```

- Note:
  - — Whenever you PRINT @ on the bottom line of the display, there is an automatic line-feed, causing everything displayed to move up one iine. To suppress this, use a trailing semicolon at the end of the statement.
  - — Example

    ```
    1ØØ PRINT @ 1ØØØ, 1ØØØ;
    ```

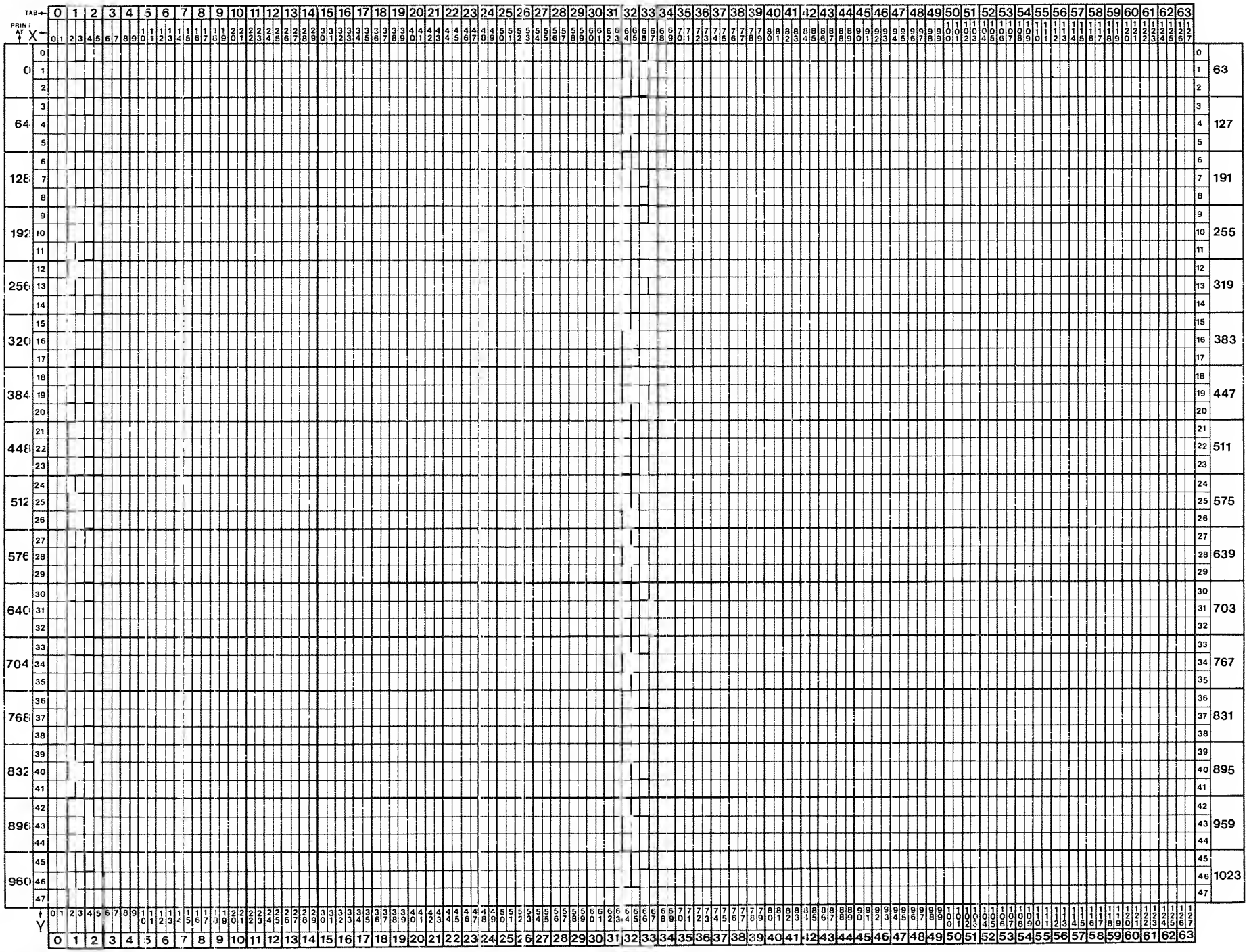
- Warning:
  - — Be sure not to use shifted @ (that is, pressing the SHIFT key with the @ key). This will cause a syntax error.

Point (X, Y)

- Asks a question
  - — Is POINT (X, Y) lit?
  - — If yes, the POINT statement prints " –1."
  - — If no, the POINT statement prints "Ø."
- Example
  - — PRINT POINT (3Ø, 3Ø)
    Since POINT (3Ø, 3Ø) was not lit, the answer should come back "Ø."
  - — Set (3Ø, 3Ø): PRINT POINT (3Ø, 3Ø)
    Since POINT (3Ø, 3Ø) was lit, the answer should come back " –1. "

TRS-80 Video Display Worksheet

TITLE _____ PROGRAMMER _____ COMMENTS _____ PAGE ____ OF ____

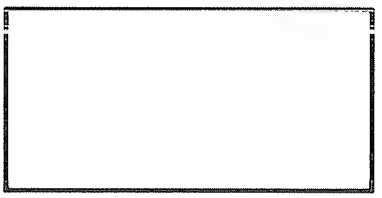


In-Class Exercise 12-3

```
10 REM * LIGHT PROGRAM *
20 INPUT "DO YOU WISH TO LIGHT THE BLOCK (IF YES, TYPE 1; IF NO, TYPE 0)";K
30 CLS
40 IF K = 0 GOTO 70
50 SET (65, 25)
60 GOTO 80
70 RESET (65, 25)
80 IF POINT (65, 25) = -1 PRINT @ 200, 65, 25, "IS ON"
90 IF POINT (65, 25) = 0 PRINT @ 200, 65,25, "IS OFF"
99 GOTO 99
```

The above program lights up a spot on the screen and then interrogates that point. RUN the program several times and see what happens. Answer either yes (1) or no (0), then follow the program action to see what is happening.

In-Class Exercise 12-4

ACTION	DISPLAY
1. Type and ENTER. ⟶	1Ø CLS 2Ø FOR J = Ø TO 127 3Ø SET (J,Ø) 4Ø NEXT J 5Ø FOR K = Ø TO 47 6Ø SET (127,K) 7Ø NEXT K 8Ø FOR L = 127 TO Ø STEP - 1 9Ø SET (L,47) 1ØØ NEXT L 11Ø FOR M = 47 TO Ø STEP - 1 12Ø SET (Ø,M) 13Ø NEXT M 14Ø FOR N = 1 TO 15ØØ 15Ø NEXT N 999 GOTO 999
2. Type RUN and ENTER.	

In-Class Exercise 12-4 (Questions)

1. Line 1Ø _____ the screen.

2. Lines 2Ø through 4Ø draw a _____ line across the _____ of the screen. (vertical, horizontal) (top, bottom)

3. Lines 5Ø through 7Ø draw a _____ line down the _____ side of the screen. (vertical, horizontal) (right, left)

4. Lines 8Ø through 1ØØ draw a _____ line from _____ to (vertical, horizontal) (right, left) _____ across the _____ of screen. (right, left) (bottom, top)

5. Lines 11Ø through 13Ø draw a line from _____ of _____ side of the screen. (top, bottom) (right, left)

6. Lines 14Ø through 15Ø are used to count from _____ to _____to _____ the screen display. (stop, move)

7. Line 999's function is to avoid destroying the ready and prompt (>) in the _____ left-hand corner of the screen by not returning the (upper, lower) control to prompt. (Note! Delete Line 999 and then run the program again. What happened?)

In-Class Exercise 12-5

ACTION	DISPLAY
1. Type and ENTER . ⟶	1Ø CLS 2Ø FOR J = 3Ø TO 45 3Ø FOR K = 2Ø TO 3Ø 4Ø SET (J,K) 5Ø NEXT K 6Ø NEXT J 7Ø FOR J = 34 TO 45 8Ø FOR K = 22 TO 28 9Ø RESET (J,K) 1ØØ NEXT K 11Ø NEXT J
2. Type RUN and ENTER .	

3. Modify the above program to draw several other letters.

In-Class Exercise 12-5 (Questions)

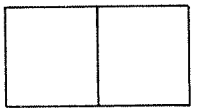
1. Line 1Ø _____ screen.
2. Lines 2Ø through 6Ø draw a lighted box starting at X = _____ and Y = _____ and ending at X = _____ and Y = _____.
3. Lines 7Ø through 11Ø erase (darken) the inside of the box starting at X = _____ and Y = _____ and ending at X = _____ and Y = _____.
4. The final letter is a _____.

PRACTICE 21

Graphics

1. Write a program that will do the following:
   a. Draw a horizontal line across the top of the screen (Line Ø).
   b. Add the necessary steps to your program to draw a vertical line down the middle of the screen.
   c. Add the necessary steps to your program to draw a horizontal line across the bottom of the screen (last line of the display).
   d. Add the necessary steps to draw a vertical line on the far left side of the display.
   e. Add the necessary steps to draw a vertical line to the far right side of the display.
   f. ENTER and RUN your program.

Display should look like this after part (E).



PART 13

Arrays

What You Will Learn

1. To explain the purpose of using arrays.
2. To set up one- and two-dimensional numeric arrays.
3. To explain the purpose and use of the terms DIM, A(3), A(2,3), DIM A(10), DIM DB(7,5).
4. To develop, enter, and run programs using numeric arrays.

Arrays

A. What is an array?

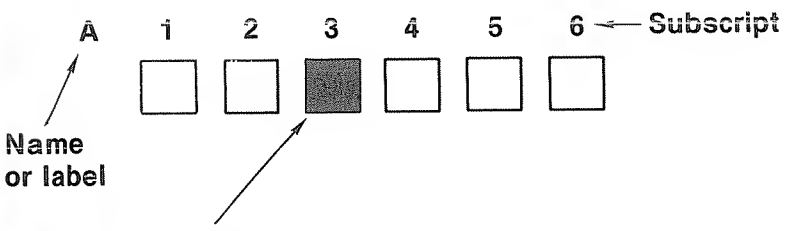
- An array is a lineup, an arrangement, or an orderly grouping of things.

B. Why use an array?

- Use it when we wish to have more variables available in a program.
  - — Although the TRS-80 Level II BASIC permits the use of approximately 900 variables for numerics, sometimes thousands of variables are required for storing and retrieving many pieces of data.
  - — The array allows you to arrange your data so that it can be stored and retrieved easily.

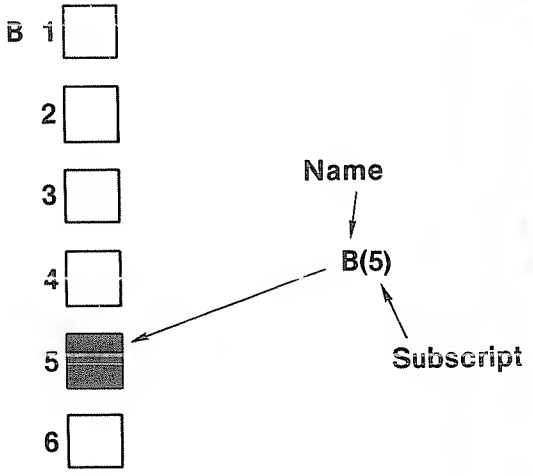
One-Dimensional Array — Illustration

SIX-ELEMENT ARRAY — NAMED A*



- A(3) is pronounced A SUB 3.
  - — A(3) represents the third cell or box in the array (lineup).
  - — Data stored in this cell would be addressed by the label A(3).
  - — Suppose data were stored in the sixth cell: A(6)? (You got it!)

SIX-ELEMENT ARRAY — NAMED B*



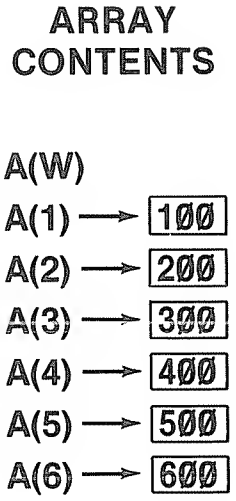
- B(5) represents the fifth cell in the array where data can be stored and retrieved.

*A and B are optional names. Any valid variable name can be used to name an array in Level II BASIC.

One-Dimensional Array — Program Example

PROGRAM	DISPLAY	REMARKS
10 DATA 100, 200, 300, 400, 500, 600 20 FOR W = 1 TO 6 30 READ A(W) 40 NEXT W		• Lines 20-40 store data in array A(W)
50 FOR W = 1 TO 6 60 PRINT W, A(W) 70 NEXT W RUN	1 100 2 200 3 300 4 400 5 500 6 600	• Lines 50-70 retrieve data from array A(W)

One-Dimensional Array — Program Example (Con't)



Above is an illustration of what happens after data are stored in array A(W). Note that in location A(1), the first data element (100) is stored. In location A(2), the second data element (200) is stored, and so on until the sixth data element (600) is stored in location A(6). Remember that line 10 of the program contained the data elements that were read using lines 20 through 40.

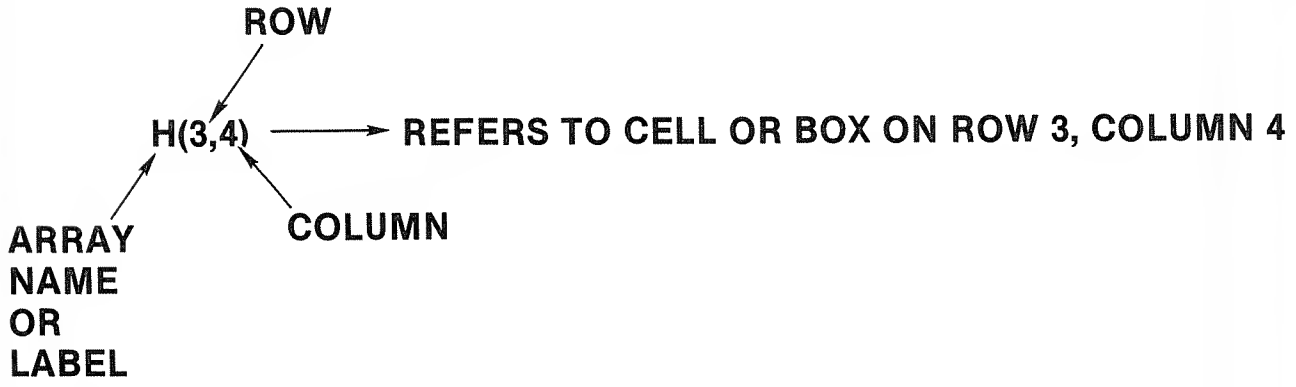
Two-Dimensional Array — Illustration

COLUMN

ROW

H	1	2	3	4	5	6
1	11	12	13	14	15	16
2	21	22	23	24	25	26
3	31	32	33	34	35	36
4	41	42	43	44	45	46
5	51	52	53	54	55	56
6	61	62	63	64	65	66

36 ELEMENT ARRAY (MATRIX)
(NAMED H)



In-Class Exercise 13-1

(Fill in the Blanks Using the Matrix Slide 13-7)

LABEL	ROW	COLUMN	CONTENTS
H(1,1)	_____	_____	_____
H(4,5)	_____	_____	_____
H(3,3)	_____	_____	_____
H(2,3)	_____	_____	_____
H(6,6)	_____	_____	_____
H(1,6)	_____	_____	_____
H(2,4)	_____	_____	_____
H(4,4)	_____	_____	_____

DIM Statement

- DIM - Lets you set the depth (number of elements allowed per dimension)
  - — If no DIM statement is used, a depth of 11 (subscripts Ø-1Ø) is allowed for each dimension of each array used.
  - — DIM statements may be placed anywhere in your program.
  - — To redimension an array, you must first use a CLEAR statement. Otherwise, an error will result!
- EXAMPLE

```
1Ø  DIM  A(6),        B(2,3),        C(21)
```

Sets a one-dimension array A with 6 elements
A(Ø) — A(5)
or
A(1) — A(6)*

Sets a two-dimension array B with 3 ROWS (numbered Ø-2) and 4 COLUMNS (numbered 0-3)

Sets a one-dimension array with 21 elements
A(Ø) — A(2Ø) or A(1) — A(21)*

Checkbook Array Example

- Consider the following table of checkbook information:

Check #	Date Written	Amount
1ØØ	6/5/81	$ 15.5Ø
1Ø1	6/7/81	25.ØØ
1Ø2	6/15/81	145.ØØ
1Ø3	6/22/81	65.ØØ
1Ø4	6/3Ø/81	211.ØØ
1Ø5	6/3Ø/81	79.5Ø

- Note that every item in the table may be specified by reference to two numbers: the row number and the column number. For example, (Row 3, Column 3) refers to the amount $145.ØØ.
- The above table can be set up in a 6 × 3 array or matrix (see next page).

Checkbook Array Example (Con't)

CK	1	2	3
1	1ØØ	6Ø581	15.5Ø
2	1Ø1	6Ø781	25.ØØ
3	1Ø2	61581	145.ØØ
4	1Ø3	62281	65.ØØ
5	1Ø4	63Ø81	211.ØØ
6	1Ø5	63Ø81	79.5Ø

6 × 3 MATRIX (ARRAY) — NAMED CK

NOTES:

1. Data recorded in form mm ddyy where mm = month number, dd = day, and yy = last two digits of year.
2. Since CK is a numeric array, alpha-numerical characters such as dashes cannot be stored.

Checkbook Array Example (Con't)

YOUR ACTION **DISPLAY**

1. Setting Up the Array
(Lines 1Ø through 11Ø)

A. Let's type and ENTER Lines 1Ø through 11Ø as shown:

(NOTE: Line 1Ø sets up dimension of array. Lines 2Ø-11Ø read the values into array CK.)

NOTE: DIM CK (6, 3) Sets up a 6 × 3 array (excluding zero subscripts) with 6 rows (numbered 1 to 6) and 3 columns (numbered 1 to 3)

```
1Ø DIM CK (6,3)
2Ø FOR ROW = 1 TO 6
3Ø FOR COL = 1 TO 3
4Ø READ CK(ROW, COL)
5Ø NEXT COL, ROW
6Ø DATA 1ØØ, 6Ø581, 15.5Ø
7Ø DATA 1Ø1, 6Ø781, 25.ØØ
8Ø DATA 1Ø2, 61581, 145.ØØ
9Ø DATA 1Ø3, 62281,  65.ØØ
1ØØ DATA 1Ø4, 63Ø81, 211.ØØ
11Ø DATA 1Ø5, 63Ø81,  79.5Ø
```

2. Manipulating the Array
(Finding the Sum)

A. Add lines 12Ø through 16Ø to the program as shown:

(NOTE: Lines 12Ø-16Ø add up all the checks written.)

```
12Ø FOR ROW = 1 TO 6
13Ø SUM = SUM + CK (ROW, 3)
14Ø NEXT ROW
15Ø PRINT "TOTAL OF CHECKS WRITTEN";
16Ø PRINT USING "$$###.##"; SUM
```

B. Type RUN and ENTER.

```
TOTAL OF CHECKS WRITTEN $541
```

NOTE: ROW and COL are used for convenience. Remember, however, the computer will only use the first two characters, RO and CO in this example.

Checkbook Array Example (Con't)

YOUR ACTION **DISPLAY**

3. Manipulating the Array (Print out all checks written on a given day)

A. Do not type NEW.

B. Add the following steps to your program: →

```
200 INPUT "LIST CHECKS WRITTEN ON (MM DD YY)": DT
210 PRINT: PRINT "CHECKS WRITTEN ON"; DT; "ARE LISTED BELOW:"
215 PRINT
220 PRINT "CHECK #", "AMOUNT": PRINT
230 FOR ROW = 1 TO 6
240 IF CK (ROW,2) = DT PRINT CK (ROW,1), CK (ROW,3)
250 NEXT
```

C. Type RUN and ENTER.

```
TOTAL OF CHECKS WRITTEN $541
LIST CHECKS WRITTEN ON (MM DDYY)?
```

D. ENTER a date (e.g., 63081 which is 6/30/81). →

```
CHECKS WRITTEN ON 63081 ARE LISTED BELOW;
CHECK #      AMOUNT
 104          211
 105          79.5
```

Assignment 13-1

1. Read Chapter 21 (pp. 123-128) TRS-80 Level I BASIC manual.
   A. Do Exercise 21-1 on page 128.
2. Read Chapter 6 (page 6/1-6/3) TRS-80 Level II BASIC Reference manual.

Summary

- A2 ≠ A(2)
  - — A2 is an ordinary variable
  - — A(2) is a subscripted variable
- Any time you have a subscript larger than 10 (depth of 11), you must use a DIM statement.
  - — Example:

    1Ø DIMA (25), B(17, 18)
- One-Dimensional Array

  SUBSCRIPT

  — A(3) is pronounced A SUB 3

  NAME
- Two-Dimensional Array (Matrix)

  ROW

  — H(3,4) refers to cell or box on row 3, column 4

  NAME COLUMN

PRACTICE 22

Arrays

1. Write a program to read the following numbers into an array and then PRINT them out:
   676 15Ø 175 188 19Ø 277 876 976 912 544
2. Change program to find the sum and average of the 1Ø numbers given.
3. Label the answer: The sum is _____, and the average is _____.

PRACTICE 23

One-Dimensional Array

1. Suppose we had the following results of a quiz given to a class of 1Ø students:

Student #	1	2	3	4	5	6	7	8	9	1Ø
Student's Grade	75	85	95	87	1ØØ	77	83	69	98	88

   a. Using a one-dimensional Array, write a program to find the class average.
   b. Add the necessary program lines to find the highest grade and the lowest grade.
   c. Have the program PRINT : Class Average is _____, Highest Grade is _____ and Lowest Grade is _____.
   d. ENTER and RUN each of these programs several times.

PART 14

INT(X), ABS(X), & RND(X) Functions

What You Will Learn

1. To explain the purpose and use of INT(X), ABS(X), and RND(X) functions.
2. To explain the purpose and use of the terms RANDOM, PSEUDO-RANDOM, SEED, RESEED.
3. To write, run, and analyze programs using the INT(X), ABS(X), and RND(X) functions.

INT(X) Function

- INT(X) or integer function allows you to round off any number, large or small, positive or negative, into a whole number (or integer).
- INT(X) means
  - — If X is a positive number, then the largest whole number can be found by chopping off the decimal part.

    Example:

    INT (5.7) = 5

    INT (Ø.7) = Ø
  - — If X is a negative number, the largest whole number can be found by moving down to the next lowest whole number (that is, make a negative number more negative).

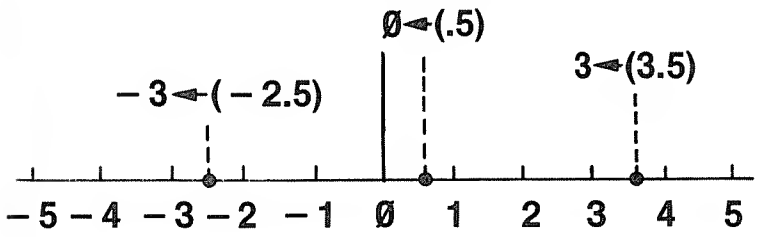
    Examples:

    INT (− . 6) = − 1 INT (− 3.14) = − 4

    INT (− .2) = − 1 INT (− 7.28) = − 8

Exercise 14-1 INT(X)

Graphical Representation



For negative numbers:
"Move to next lowest whole number"

For positive numbers:
"Chop off decimal part"

X	INT(X)
Ø.5	_____
– 1.7	_____
2.345	_____
– Ø.8	_____
Ø	_____
3.1415	_____
76.14	_____
– 1Ø.35	_____

INT(X) FUNCTION — ROUNDING $$

YOUR ACTION	DISPLAY
1. Type and ENTER this program. →	10 LET A = 20/3
2. Now RUN.	20 PRINT "$"; A $ 6.66667
3. Add Line 15 to program as shown. (Note: In Line 15 we multiply by 100, add .5, take the INT, which is now 667, and then divide 667 by 100. 667/100 is 6.67, which is what we want, two decimal places.)	15 A = INT (100*6.66667+.5)/100
4. Now RUN program. (I told you so!)	$6.67

Assignment 14-1 INT(X)

1. Type NEW and [ENTER] this program for finding the area of a circle:

```
1Ø REM** AREA OF A CIRCLE 3.14159* R ↑ 2 **
2Ø INPUT "THE RADIUS IS"; R
3Ø P = 3.14159
4Ø A = P*R ↑ 2
5Ø PRINT "THE AREA IS"; A
```

2. RUN the program several times to make sure it works.
3. Change the program to suppress (chop off) all of the numbers to the right of the decimal point. (RUN the program to make sure it works.)
4. Change the program to make the answer accurate to one decimal place. (For example, if R = 1, then Area (A) = 3.1.)

ABS(X) Function

- ABS(X) = Abbreviation for absolute value of X
- Examples:

  ABS (12) = 12 ABS (−10) = 10
  ABS (0) = 0 ABS (−357) = 357

- Note! ABS (25 − 10) = ABS (10 − 25) = 15

Assignment 14-2 ABS(X)

YOUR ACTION	DISPLAY
1. Type and ENTER the program shown.	10 INPUT "TYPE ANY POSITIVE OR NEGATIVE #"; N 20 X = ABS (N) 30 PRINT "N", "X" 40 PRINT N,X
2. RUN the program several times using both positive and negative numbers. (Note that regardless of the number you input as N, the absolute value of X is the same number without the sign.)	

RND(X) Function

- RND(X) or random number function causes the computer to give you a "surprise" number.
  - — It's as though the computer spins a wheel of chance.
  - — It's like pulling a number out of a hat.
  - — It's unpredictable!
- The random number function – general form

  Let N = RND(X)

  Where N = The random number

  RND = Abbreviation for random

  X = Any number between 1 and 32767
- Random* is a complete program statement that "reseeds" the random number generator.
  - — You may want to put random at the beginning of a program to ensure you don't get the same random number sequence each time you turn the computer on to play a game.

*NOTE: Since random numbers are unpredictable and computers are not, we get pseudorandom (i.e., not the real thing!) numbers. Each time you use RND function the computer uses an internal seed number to produce the desired random number. Therefore, it is a good idea to set the seed number (reseed) to an unpredictable value when you are running game programs.

Random Number — Program Example

YOUR ACTION	DISPLAY
1. Type and ENTER. (Line 5 reseeds the random number generator. This line is an option in the program.)	5 RANDOM 1Ø FOR N = 1 TO 6 2Ø PRINT RND (Ø); 3Ø NEXT N
2. RUN and ENTER. (Observe that there are six different numbers between Ø and 1.)	(SCREEN SHOULD HAVE SIX RANDOM NUMBERS BETWEEN Ø AND 1.)
3. RUN program again to get the idea.	
4. Change Line 1Ø to read:	1Ø FOR N = 1 TO 1ØØ
5. RUN. (Get the idea?)	(SCREEN SHOULD BE FILLED WITH RANDOM NUMBERS BETWEEN Ø AND 1.)

Coin Toss Program

ACTION AND REMARKS	DISPLAY
1. Type and [ENTER] program as shown:	5 REM**COIN TOSS PROGRAM**
	1Ø REM**H = HEADS, T = TAILS**
	2Ø H = Ø: T = Ø: PRINT
	3Ø INPUT "HOW MANY TIMES SHALL I FLIP THE COIN"; N
	4Ø CLS
	5Ø PRINT "I'M FLIPPING THE COIN... STANDBY"
	6Ø FOR K = 1 TO N
(Line 2Ø initializes counters, sets H = T = Ø.)	
(Line 4Ø starts next line at top of screen.)	
(Line 6Ø begins FOR-NEXT statement and runs it "N" times.)	
(Line 7Ø generates integers between 1 and 2.)	7Ø X = RND (2)
(Line 8Ø tells the program to go to Line 9Ø if X = 1 = heads and to Line 1ØØ if X = 2 = tails.)	8Ø ON X GOTO 9Ø, 1ØØ
(Line 9Ø, "heads" are counted.)	9Ø H = H + 1 :GDTO 11Ø
(Line 1ØØ, "tails" are counted.)	1ØØ T = T + 1
(Line 11Ø sends control back to Line 6Ø for "N" passes.)	11Ø NEXT K
	12Ø CLS

Coin Toss Program (Con't)

ACTION AND REMARKS	DISPLAY
(Line 13Ø prints the headings.)	13Ø PRINT "HEADS", "TAILS", "TOTAL FLIPS" : PRINT
(Line 14Ø prints the values of H, T, and N.)	14Ø PRINT H, T, N
(Line 15Ø calculates and prints the percentage of heads, percentage of tails.)	15Ø PRINT 1ØØ*H/N; "%", 1ØØ*T/N; "%"
(Line 16Ø provides spacing for better appearance.)	16Ø PRINT: PRINT: PRINT

Assignment 14-3 RND(X)

YOUR ACTION	DISPLAY
1. Type and [ENTER] the program as shown. →	5 REM ** PICK A NUMBER GAME ** 1Ø RANDOM 2Ø X = RND (1Ø) 3Ø INPUT "ENTER A NUMBER BETWEEN 1 & 1Ø"; N 4Ø IF X = N THEN 1ØØ 5Ø IF X < N THEN 11Ø 6Ø IF X > N THEN 12Ø 1ØØ PRINT "RIGHT ON" 1Ø5 FOR J = 1 TO 25ØØ : NEXT : GOTO 1Ø 11Ø PRINT "LOWER" : GOTO 3Ø 12Ø PRINT "HIGHER" : GOTO 3Ø
2. RUN the program.	

Assignment 14-3 RND(X)

3. Analyze the program.

   Line 1Ø _____ the random number generator.

   Line 2Ø is the _____ generator.

   Line 3Ø allows the user to _____ a number.

   Lines 4Ø, 5Ø, and 6Ø are _____ statements that compare
   conditional, unconditional

   the random number _____ with the input number _____.
   X,N X,N

   Lines 1ØØ, 11Ø, and 12Ø are PRINT statements that guide the player. Why does Line 1ØØ GOTO Line 1Ø and why do Lines 11Ø and 12Ø GOTO Line 3Ø?

4. Modify (change) the program to pick a number between 1 and 1ØØ, and RUN this program several times.

Summary

- ABS(X) — Provides the absolute value of X regardless of the number you input (i.e., X is that same number without the sign).
- INT(X) — Provides integer or whole number value of X.
  - — If X is a positive (+) number, it chops off the decimal part.
  - — If X is a negative number, it rounds down to the next lowest whole number (e.g., INT (−Ø.6) = −1).
- RND(X) — Causes the computer to give you a random number.
  - — RND (Ø) is for random numbers greater than Ø and less than 1.
  - — RND (N) is for random numbers from 1 to N.
- RANDOM — is a complete program statement that reseeds the random generator.
  - — It should be used at the beginning of game programs to ensure you will not get the same pseudo random number sequence each time.
  - — Pseudorandom essentially means that, for all practical purposes, the numbers will be random or unpredictable.

PRACTICE 24

INT(X) and ABS(X)

1. Fill in the blanks with the appropriate INT(X):

X	INT(X)
0.7	_____
−2.5	_____
6.365	_____
−0.8	_____
−10.65	_____
0	_____
3.2425	_____
−7.61	_____
−0.3	_____
0.3	_____

2. The following program can be used for finding the area of a circle:

```
10 REM *** AREA OF A CIRCLE = 3.14159 * R ↑ 2 ***
20 INPUT "THE RADIUS IS"; R
25 INPUT "THE RADIUS IS IN (IN.,FT,OR YD,)"; A$
30 A = 3.14159 * R ↑ 2
40 PRINT "THE AREA IS"; A; "SQ"; A$
```

   a. ENTER and RUN the program several times to make certain it works.
   b. Change the program to suppress (chop off) all the numbers to the right of the decimal point (RUN the program to make sure it works).
   c. Change the program to make the answer accurate to one decimal place. (For example if R = 1, then area (A) = 3.1.)

PRACTICE 25

Random Number

1. Write a program that will let you pick a random number between 1 and 100. The program should input a number from the keyboard and provide the following clues on your guess.
   a. If the number you pick matches the number the computer picks, have the computer PRINT "Right On."
   b. If the number from the keyboard is too high, have the program print "Lower."
   c. If the number from the keyboard is too low, have the program print "Higher."
   d. ENTER and RUN the program several times.

PART 15

Subroutines

What You Will Learn

1. To explain the purpose for using subroutines.
2. To explain the purpose and use of terms ON-GOTO, GOSUB, RETURN, ON-GOSUB.
3. To develop, enter, and run programs using subroutines and ON-GOTO statements.

Subroutine

What Is It?

- A subroutine is a short program or routine that is built into a large program to do specific calculations or perform repetitive functions.

Why Use It?

- There are times when you need the same type of calculation at various points in your program, but instead of retyping the statements needed for this calculation each time, you can write a subroutine to perform the needed calculations.

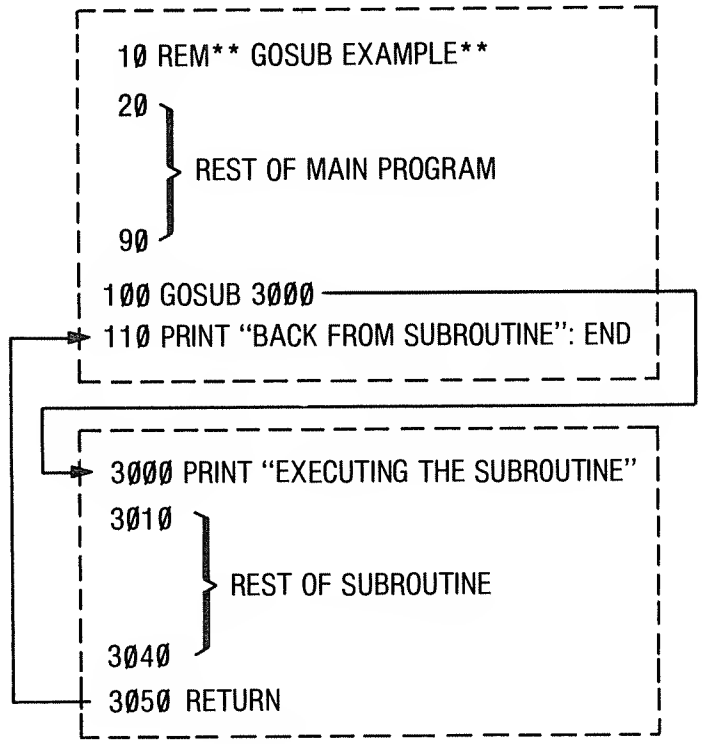
How Do You Call a Subroutine?

- To call or branch to a subroutine, use the GOSUB statement.
  - The GOSUB XXXXX statement directs the computer to go to that line number and execute the program steps until it reaches the key word RETURN, which ends the subroutine.
  - RETURN is always built into a subroutine and is used to tell the computer that the subroutine is finished. When finished, the control of the program is returned to the statement in the main program immediately following the most recently executed GOSUB.

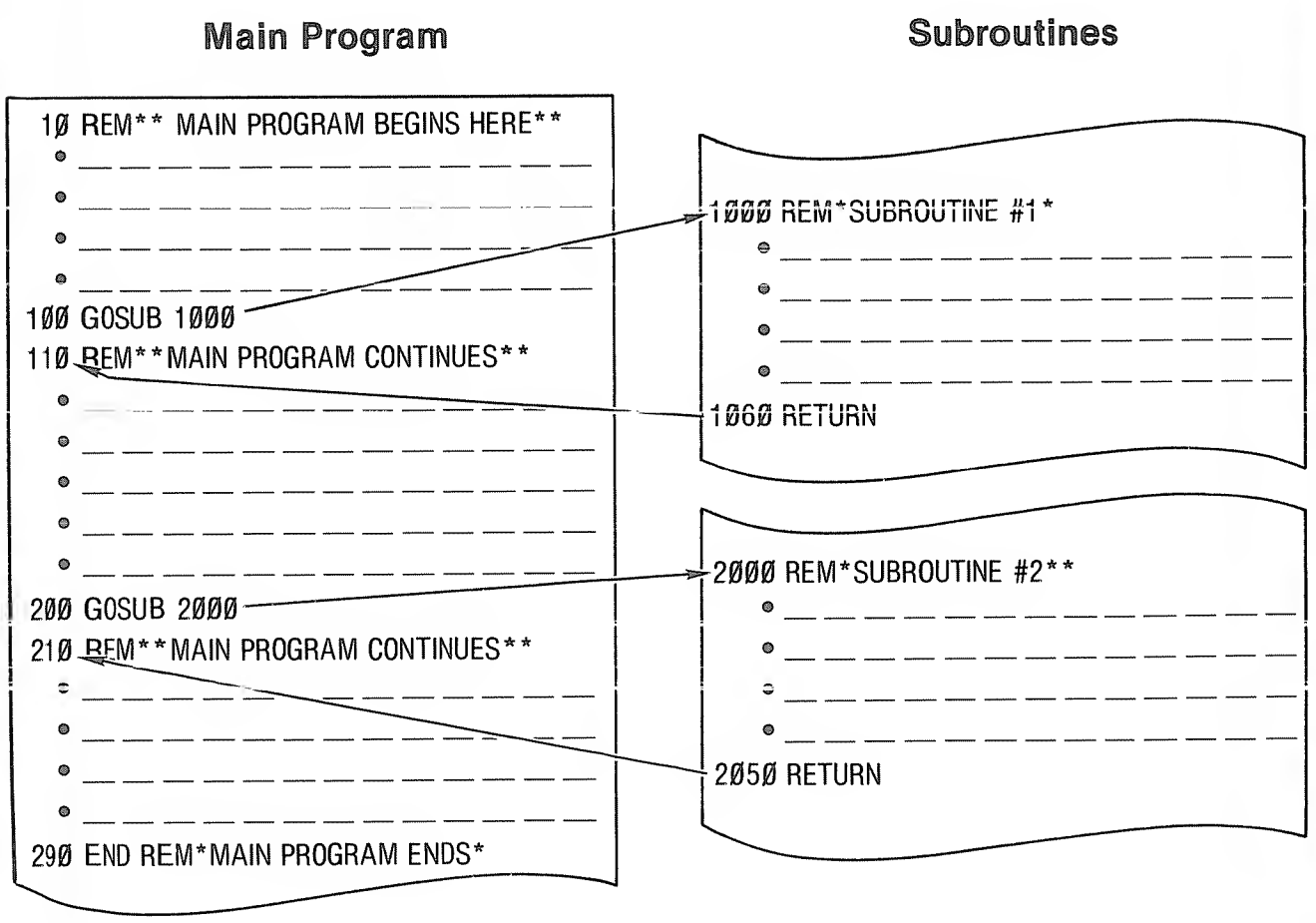
Subroutine Example

Main Program:

Subroutine:



Subroutine Illustration



Subroutine Illustration (Con't)

1. When the computer reaches the GOSUB in Line 1ØØ, the program will branch (GOTO) Line 1ØØØ, which is the beginning of Subroutine #1.
2. After Subroutine #1 is executed and the RETURN (Line 1Ø6Ø) is reached, control is passed back to the main program (Line 11Ø). Note that Line 11Ø is the next higher number after the GOSUB that put it in the subroutine (Line 1ØØ).
3. The computer continues through the main program to the GOSUB in Line 2ØØ, which branches control to Subroutine #2 in Line 2ØØØ.
4. After the subroutine is executed, the RETURN (Line 2Ø5Ø) passes the control back to Line 21Ø in the main program. (Note again that this is the next higher line number after the GOSUB in Line 2ØØ.)
5. An END statement is included in the program (Line 29Ø) after the main program is finished to keep it from accidentally falling into the subroutine. We only want the subroutines to be executed when we call for them by a GOSUB.

Sample Program Using Subroutines (Temperature Conversion)

Main Program

```
 1Ø REM*TEMPERATURE CONVERSION PROGRAM**
 15 CLS
 2Ø INPUT "DO YOU WISH TO CONVERT C TO F (Y OR N)"; A$
 3Ø IF A$ = "Y" THEN 8Ø ELSE PRINT
 4Ø PRINT: INPUT "DEGREES FAHRENHEIT"; F
 5Ø GOSUB 2ØØØ
 6Ø PRINT: INPUT "HAVE YOU FINISHED (Y OR N)"; B$
 7Ø IF B$ = "N" THEN 4Ø ELSE END
 8Ø PRINT: INPUT "DEGREES CENTIGRADE"; C
 9Ø GOSUB 1ØØØ
1ØØ PRINT: INPUT "HAVE YOU FINISHED (Y OR N)"; C$
11Ø IF C$ = "N" THEN 8Ø ELSE END
12Ø END
```

Subroutine #1

```
1ØØØ REM**CELSIUS TO FAHRENHEIT CONVERSION**
1Ø1Ø F = (9/5)* C + 32 : PRINT
1Ø2Ø PRINT C; "DEGREES CELSIUS ="; F; "DEGREES FAHRENHEIT"
1Ø3Ø RETURN
```

Subroutine #2

```
2ØØØ REM**FAHRENHEIT TO CELSIUS CONVERSION**
2Ø1Ø C = (F-32) * (5/9): PRINT
2Ø2Ø  PRINT F; "DEGREES FAHRENHEIT ="; C; "DEGREES CELSIUS"
2Ø3Ø RETURN
```

Analysis of Sample Program Using Subroutines

1. Lines 1Ø through 11Ø comprise the main program.
2. Line 2Ø is an input statement to ask the user if he wants to convert from C to F or from F to C. Yes (Y) means C to F and No (N) means F to C.
3. Line 3Ø is a conditional branch statement. If the user wants to convert Centigrade C to Fahrenheit, then branch to Line 8Ø; otherwise, skip a line (PRINT) and go to Line 4Ø.
4. Line 4Ø allows the user to input the °F to be converted to °C.
5. Lines 5Ø and 9Ø call the subroutines.
6. Line 6Ø asks the user if he is finished. In Line 7Ø the program will branch to Line 4Ø (if B$ = N) or the program will END (if B$ ≠ N).
7. Line 8Ø is similar to Line 4Ø, except that it allows the user to input the °C to be converted to °F.
8. Lines 1ØØ and 11Ø are the same as Lines 6Ø and 7Ø.
9. The first subroutine begins at Line 1ØØØ and ENDS at Line 1Ø3Ø. It RETURNS control to Line 1ØØ in the main program.
10. The second subroutine begins at Line 2ØØØ and ENDS at Line 2Ø3Ø. It RETURNS control to Line 6Ø in the main program.

Subroutine Exercise

```
  1Ø PRINT "THIS IS"; "  ";
  2Ø GOSUB 1ØØØ
  3Ø PRINT "OF HOW"; "  ";
  4Ø GOSUB 2ØØØ
  5Ø PRINT "WORKS"
  6Ø END
1ØØØ PRINT "AN EXAMPLE"; "  ";
1Ø1Ø RETURN
2ØØØ PRINT "A SUBROUTINE"; "  ";
2Ø1Ø RETURN
```

1. Analyze the program and write the message. _____
2. Now type and [ENTER] the program.
3. RUN the program. Does it agree with your message?

Assignment 15-1

1. Analyze the program below and write the message:

```
10 LET B = 10
20 GOSUB 2000
30 B = B + 5
40 GOSUB 2000
50 B = B + 10
60 GOSUB 2000
99 END
2000 REM SUBROUTINE
2010 IF B<12 THEN 2050
2020 IF B = 25 THEN 2070
2030 PRINT "PRIME"
2040 GOTO 2080
2050 CLS: PRINT "LEEDS"
2060 GOTO 2080
2070 PRINT "COMPUTERS"
2080 RETURN
```

Message _____

ON-GOTO Example

YOUR ACTION	DISPLAY
1. Type NEW and ENTER this program:	5 CLS 1Ø INPUT "TYPE A NUMBER FROM 1 TO 3", N 2Ø IF N = 1 THEN 11Ø 3Ø IF N = 2 THEN 13Ø 4Ø IF N = 3 THEN 15Ø 5Ø PRINT "HEY, I WANT A NUMBER FROM 1 TO 3!" 6Ø GOTO 1Ø 99 END 11Ø PRINT "N = 1" 12Ø END 13Ø PRINT "N = 2" 14Ø END 15Ø PRINT "N = 3" 16Ø END
2. Before you RUN the program, analyze it. Can you predict what will happen when you RUN it?	
3. RUN the program several times until you feel comfortable with it.	

ON-GOTO Example (Con't)

YOUR ACTION	DISPLAY
4. Erase Lines 2Ø, 3Ø, and 4Ø. (Remember, there are two ways to do this! Use DELETE or simply type in each line number separately and then press [ENTER].)	
5. Type and [ENTER] this line:	2Ø ON N GOTO 11Ø, 13Ø, 15Ø
6. List your program.	(SHOULD HAVE NEW LINE 2Ø + LINES 5, 1Ø, AND 5Ø THRDUGH 16Ø FRDM PREVIOUS PAGE. IF YOU DON'T HAVE THESE LINES, FIX IT!) (WORKS JUST THE SAME AS BEFORE, DDESN'T IT?)
7. RUN the program a few times.	
8. RUN the program again. Use the following inputs: **1.5** **1.8** **2.8** **Ø.8** **3.99**	N = 1 N = 1 N = 2 HEY, I WANT A NUMBER BETWEEN 1 & 3! N = 3
(Now do you understand that N = INT (N) or whole number?)	

ON-GOTO Example Analysis

1. Line 2Ø tells the computer to do the following:
   - If, the integer (whole number) value of N is 1, GOTO Line 11Ø.
   - If the integer value of N is 2, GOTO Line 12Ø.
   - If the integer value of N is 3, GOTO Line 13Ø.
   - If the integer value of N is not one of the numbers listed above, then move on to the next line.
2. The ON-GOTO statement has a built-in INT statement, which really acts like this:
   2Ø ON INT (N) GOTO----ETC.

Assignment 15-2 ON-GOTO

1. Type and [ENTER] the following program:

```
 5 CLS
 10 INPUT "ENTER A NUMBER FROM 1 TO 5"; N
 20 ON N GOTO 100, 200, 300, 400, 500
 30 PRINT "HEY I WANT A NUMBER FROM 1 TO 5!": GOTO 10
 40 END
100 PRINT "N = 1": END
200 PRINT "N = 2": END
300 PRINT "N = 3": END
400 PRINT "N = 4": END
500 PRINT "N = 5": END
```

2. Answer the following questions before running the program
   a. What happens (output) if the input is 1.8 (Line 10)?_____
   b. What happens (output) if the input is 3.99? _____
   c. What happens (output) if the input is 2.89? _____
   d. What happens if the input is 0.5? _____

3. RUN the program several times and record the following:

INPUT	OUTPUT

ON-GOSUB

- Works like ON-GOTO, except control branches to one of the subroutines specified by the line numbers in the line number list.
- Example:

```
10 INPUT "CHOOSE 1, 2, OR 3"; K
20 ON K GOSUB 1000, 2000, 3000
99 END
1000 PRINT "SUBROUTINE #1" : RETURN
2000 PRINT "SUBROUTINE #2" : RETURN
3000 PRINT "SUBROUTINE #3" : RETURN
```

- K may be a numerical constant, variable, or expression.
  - — It must have a positive value, however, or an error will occur.
- If K ≠ 1, 2, or 3, the program will go to the next line (99 END).

Summary

- GOSUB XXXX, causes the computer to:
  - — Go to the subroutine beginning at line XXXX (the specified line number).
  - — Work through the subroutine until it finds a RETURN statement.
  - — Return control to the statement that follows the GOSUB statement in the main program.
- ON n GOSUB XXXX, ------, YYYY
  - — Multi-way branching statement that is controlled by a test variable (n), which sends control of the program to one of the subroutines specified by line numbers in the line number list (i. e., XXXX,----, YYYY).
  - — The test variable n must be a numerical constant, variable, or expression that has a non-negative value or else an error will occur.
- ON n GOTO XXXX, ----, YYYY
  - — Works like ON n GOSUB except control branches to one of the line numbers specified (XXXX, ----, YYYY).
  - — ON n GOTO 1st line number, 2nd line number ———— nth line number expression must be between Ø and 255 inclusive.
  - — If n<Ø, an error will occur.

PRACTICE 26

Program to Convert Centigrade to Fahrenheit and Vice Versa

1. Write a program that will do the following:
   a. Convert Centigrade to Fahrenheit.
   b. Convert Fahrenheit to Centigrade.
   c. Allow you to select either A or B above.
   d. Allow you to input from keyboard.
   e. PRINT the answer as follows:

   ___*___ degrees Celsius = ___**___ degrees Fahrenheit

   or

   ___*___ degrees Fahrenheit = ___**___ degrees Celsius

   * Keyboard input value
   ** Calculated ouput value

PRACTICE 27

Program for Sample Profit/Loss Statement

1. When a product is sold for more than it costs, the seller receives a profit. When a product is sold for less than it costs, the seller takes a loss.

   Therefore: sell price − cost = profit or loss

   If we let: S = Sell price
   C = Cost
   U = No. of units
   P = Profit
   L = Loss

   Then: P (or L) = S*U − C*U

   a. Write a program that will compute the profit or loss for a business if the sell price and cost are known. (*Note:* Program should permit you to enter cost and sell price from the keyboard.)
   b. Have the computer PRINT the following:

   NO. OF UNITS _____
   UNIT PRICE ($) _____
   UNIT COST ($) _____
   TOTAL SALES ($) _____
   TOTAL COST ($) _____
   PROFIT/LOSS ($) _____
   % OF SALES _____

   c. RUN the program several times and record your answer.

EXTRA PRACTICE 1

Programming Mathematical Operators

1: Given two numbers A=25 and B=5:
   a. Write one program that will add, subtract, divide (A/B), multiply, and square the two numbers (A and B).
   b. The answer should PRINT as shown here:
      The sum of A and B is _____ (your answer).
      The difference of A and B is _____ (your answer).
      The quotient of (A/B) is _____ (your answer).
      The product of A*B is _____ (your answer).
      The square of A is _____ (your answer).
      The square of B is _____ (your answer).

EXTRA PRACTICE 2

Finding the Average

1. Write a program to find the average of three numbers.
2. Have the program PRINT: The average is _____.
3. Add a program line to have the program PRINT the average of your # _____, your # _____, and your # _____ is your answer _____. Example: The average of 3, 4, and 8 is 5.

EXTRA PRACTICE 3

More Mathematical Operations

Write five separate programs to PRINT the answer to these problems (the answer should read 25 * 2 + 4 = 54, and so on.):

1. 25*2+4
2. $3^2 + 4 - 2$
3. 36 ÷ 4 *5
4. 28 + 4 * 6 ÷ 8
5. (18–2) ÷ 3 + 4 (6*3) + 2^3

EXTRA PRACTICE 4

Print Zones

Part I.

Write a program to PRINT the word "Leeds" in the following ways:

	ZONE 1	ZONE 2	ZONE 3	ZONE 4
1.	LEEDS	LEEDS	LEEDS	LEEDS
2.	LEEDS		LEEDS	
3.		LEEDS		LEEDS
4.		LEEDS	LEEDS	LEEDS
5.			LEEDS	LEEDS
6.				LEEDS

Part II.

Using page 68, type in the information as shown (LEEDSMIDDLELEEDS)...and so on.

1. Count the number of characters in all four zones. How many?
2. How many in zone 1 _____, zone 3 _____.

EXTRA PRACTICE 5

Area of Square and Volume of Cube

1. Write a program to solve the following problems. Label your answers.
   a. The side of a square is 27 inches. Find its area (area (A) = s^2).
   b. If the side of a cube is also 27 inches, find its volume (volume (V) = s^3).
2. Using INPUT statements, write a program to find the area of a square and volume of a cube.
   a. Solve the problems above (assume sides of square and cube are equal).
   b. Using different lengths for the side, RUN the program again (assume that the sides of the square and the cube are equal).

EXTRA PRACTICE 6

Printing Tables of Numbers, Squares, and Cubes

1. Write a program to generate the first 25 numbers and PRINT their squares on the same line.
   Example:
   ```
   1     1
   2     4
   3     9
   4    16
   and so forth
   ```
2. Write a program to generate the first 25 numbers and PRINT their cubes on the same line.
   Example:
   ```
   1     1
   2     8
   3    27
   4    64
   and so forth
   ```
3. Write a program to generate all the numbers from 20 to 1 and PRINT the numbers, their squares, cubes, and fourth powers on the same line and in four columns.
   Example:
   ```
   20    400    8000    160000
   19    361    6859    130321
   18    324    5832    104976
   and so forth
   ```

EXTRA PRACTICE 7

Printing Three Times and Nine Times Tables

1. Write a program to generate the three times table from $3 \times 1 = 3$ to $3 \times 12 = 36$. The printout should look exactly like this:
   ```
   3 * 1 = 3
   3 * 2 = 6
   3 * 3 = 9
   3 * 4 = 12
   and so forth
   ```
2. Write a program to generate the nine times table from $9 \times 1 = 9$ to $9 \times 12 = 108$.

EXTRA PRACTICE 8

Two-Dimensional Array

1. Suppose we have a class of ten students. The course grade is based upon three quizzes, and the results for the class are as follows:

Student #	1	2	3	4	5	6	7	8	9	10
Quiz #										
1	88	41	100	88	79	76	86	90	85	100
2	75	52	65	57	98	86	96	91	86	92
3	71	47	75	77	86	96	85	92	97	82

a. Write a program to PRINT the following information:

Student #	*Course Avg./Student*	
1	?	Computer calculate and PRINT average
2	?	
3	?	
4	?	

and so forth

Quiz #	*Class Avg./Quiz*	
1	?	Computer calculate and PRINT average
2	?	
3	?	

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

ISBN 0-8104-6174-9